НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу
(повна назва інституту/факультету)

Кафедра системного проектування
(повна назва кафедри)

«На правах рукопису»
УДК 004:004.453

«До захисту допущено»

Завідувач кафедри
_____ _А.І. Петренко
(підпис) (ініціали, прізвище)

“___”_____2018 р.

Магістерська дисертація

зі спеціальності (спеціалізації)__122-комп’ютерні науки та інформаційні технології (Системне проектування сервісів)_____
(код і назва спеціальності)

на тему: Багаторівневе навчання для класифікації об’єктів на множинах надвеликих масивів даних_____

Виконав: студент _6_ курсу, групи ДА-62м
(шифр групи)

_____Акимов Вадим Сергійович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник _____проф., д.т.н., Рогоза В.С._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Розробка стартап-проекту_____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Інститут/факультет ННК «Інститут прикладного системного аналізу»
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною (освітньо-науковою) програмою

Спеціальність (спеціалізація) 8.05010103 Системне проектування
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І. Петренко
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Акимову Вадиму Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації «Алгоритми багаторівневого навчання для класифікації об'єктів на множинах надвеликих масивів даних»
науковий керівник дисертації Рогоза В.С., д.т.н., проф.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження класифікаційна модель з використанням багаторівневого навчання

4. Предмет дослідження алгоритми багаторівневого навчання для побудови класифікаційних моделей

5. Перелік завдань, які потрібно розробити 1) зібрати та підготувати вхідні дані для дослідження алгоритмів класифікації об'єктів, 2) виконати огляд теоретичного базису нейронних мереж, 3) дослідити архітектуру згорткових нейронних мереж для аналізу зображень, 4) виконати огляд можливих програмних інструментів для практичної побудови та навчання нейронних мереж, 5) обґрунтувати вибір інструменту для даної задачі, 6) виконати

програмну реалізацію згорткової мережі та оцінити точність отриманої класифікаційної моделі.

6. Орієнтовний перелік публікацій

1. Akymov V. Deep learning algorithms for classification of skin diseases / Vadym Akymov.// International Scientific Journal "Internauka".–2018.–№9.

2. Акимов В. Алгоритми багаторівневого навчання для класифікації захворювань шкіри. / Вадим Акимов.// Матеріали XX всеукраїнської науково – практичної студентської конференції, 21 травня 2018, Київ, Україна: матеріали. – К. : НТУУ «КПІ», 2018. – С. 121-122.

7. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Реалізація стартап-проекту			

8. Дата видачі завдання 01.02.2018

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.02.2018	
2	Збір інформації та аналіз літератури	15.02.2018	
3	Проведення огляду теоретичного базису навчання нейронних мереж (класичний алгоритм градієнтного спуску і його модифікації: стохастичний та порційний)	28.02.2018	
4	Дослідження архітектури згорткових нейронних мереж.	11.03.2018	
5	Огляд програмних інструментів для навчання нейронних мереж, імплементації згорткової мережі	13.04.2018	
6	Оцінка точності отриманої моделі	25.04.2018	
7	Оформлення дипломної роботи	30.04.2018	
8	Отримання допуску до захисту та подача роботи в ДЕК	09.05.2018	

Студент _____ Акимов В.С.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Рогоза В.С.
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника

РЕФЕРАТ НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ

виконану на тему: Алгоритми багаторівневого навчання для класифікації об’єктів на множинах надвеликих масивів даних

студентом: Акимовим Вадимом Сергійовичем

Робота виконана на 79 сторінках, містить 13 ілюстрацій, 23 таблиць. При підготовці використовувалась література з 21 джерел.

Актуальність теми

Захворювання шкіри сьогодні належать до розповсюджених медичних проблем. Кількість таких захворювань постійно зростає, незважаючи на розвиток медичної галузі. Рак шкіри є поширеним злоякісним новоутворенням і займає друге рангове місце у структурі онкологічної захворюваності населення України. Такі хвороби діагностуються візуально, починаючи з клінічних обстежень, що можуть супроводжуватись дерматоскопічним аналізом, біопсією та гістопатологічною експертизою.

Особливий інтерес представляє автоматизована класифікація захворювань шкіри (як доброякісних, так і злоякісних) на базі зображення ураженої ділянки тіла. Глибокі згорткові нейронні мережі (Convolutional Neural Networks, CNN) показують потенціал для аналізу категорії дрібнозернистих зображень.

Мета та задачі дослідження

Метою даної роботи є розробка моделі багаторівневого навчання для побудови прототипу автоматизованої діагностичної системи. Даний прототип служить для вирішення задачі класифікації шкірних захворювань на основі зображень враженої ділянки тіла людини.

Рішення поставлених завдань та досягнуті результати

У даній роботі було представлено прототип автоматизованої діагностичної системи для вирішення задачі класифікації шкірних захворювань на основі зображень враженої ділянки тіла людини. Прототип було виконано на базі класифікаційної моделі з використанням

багаторівневого навчання, а в основі даної моделі лежить згорткова нейронна мережа. Імплементація графу обчислень виконана за допомогою мови програмування Python та програмної платформи Tensorflow. В якості функції помилки для діагностики точності отриманої моделі використовувалась перехресна ентропія. В результаті отримана висока точність класифікації підтверджує можливість впровадження даного прототипу в промислових масштабах в рамках існуючих медичних платформ, таких як eHealth.

Серед основних проблем, що виникли під час імплементації прототипу – це відсутність доступу до потужних дискретних відеокарт, що дозволяють за допомогою платформи CUDA прискорити навчання моделі, а також низька пропускна швидкість каналу призвела до проблем з часом обробки вхідних зображень навчального набору.

Об’єкт досліджень

Класифікаційна модель з використанням багаторівневого навчання.

Предмет досліджень

Алгоритми багаторівневого навчання для побудови класифікаційних моделей.

Методи досліджень

Для вирішення проблеми в даній роботі використовуються методи аналізу і синтезу, системного аналізу, порівняння, логічного узагальнення результатів.

Наукова новизна

Наукова новизна роботи полягає у створенні прототипу автоматизованої діагностичної системи на основі багаторівневого навчання для вирішення задачі класифікації захворювань шкіри, а також апробація отриманого прототипу на практиці з використанням вибірки даних, представленої міжнародною спільнотою з цифрової обробки зображень шкіри ISDIS (13791 зображень уражених ділянок шкіри, зроблених за допомогою дермоскопу).

Практичне значення одержаних результатів

Отримані результати можуть використовуватись у майбутніх дослідженнях за напрямком покращення запропонованого прототипу та класифікаційної моделі, враховуючи переваги та недоліки даних результатів. Також даний прототип може бути використаний для покращення результатів роботи існуючих діагностичних систем.

Публікації

1. Akymov V. Deep learning algorithms for classification of skin diseases / Vadym Akymov.// International Scientific Journal “Internauka”.–2018.–№9.

2. Акимов В. Алгоритми багаторівневого навчання для класифікації захворювань шкіри. / Вадим Акимов.// Матеріали XX всеукраїнської науково – практичної студентської конференції, 21 травня 2018, Київ, Україна: матеріали. – К. : НТУУ «КПІ», 2018. – С. 121-122.

Ключові слова

Алгоритми багаторівневого навчання, згорткова нейронна мережа, класифікація, захворювання шкіри.

ABSTRACT ON MASTER'S THESIS

on topic: Deep learning algorithms for objects classification problem within big data sets

student: Vadym Akymov

Work carried out on 79 pages containing 13 figures, 23 tables. The paper was written with references to 21 different sources.

Topicality

Skin diseases today are among the most common medical problems. The amount of these diseases is constantly growing, despite the development of medicine. Skin cancer is a common malignant neoplasm and has the second rank in the structure of cancer morbidity in Ukraine. Such diseases are diagnosed visually, beginning with clinical examinations, which can be accompanied by dermatoscopy analysis, biopsy and histopathological examination. Of particular interest is the automated classification of skin diseases (both benign and malignant) based on the image of the affected area of the body. Convolutional Neural Networks (CNN) show the potential for analyzing the category of fine-grained images.

Purpose

The purpose of this work is to develop a deep learning classification model for building an automated diagnostic system prototype. It serves to solve the problem of classification of skin diseases on the basis of images of the affected area of the human body.

Solution

In this paper a prototype of an automated diagnostic system was presented for solving the skin diseases classification problem based on the images of affected area of the human body. The prototype was implemented using deep learning algorithms, and convolutional neural network serves as a kernel part of the system. Computation graph is developed using the Python programming language and the Tensorflow software platform. As a function of the error to diagnose the accuracy of the model, cross-entropy was used. The resulting high classification accuracy

confirms the feasibility of implementing this prototype on an industrial scale within existing medical platforms, such as eHealth.

Among the main problems encountered during the implementation of the prototype is the lack of access to powerful discrete graphics cards, which allow CUDA platform to accelerate model learning, as well as low bandwidth of the network channel, resulting in problems with the processing time of incoming images of the training set.

Object of research

Deep learning classification model.

Subject of research

Deep learning algorithms for building classification models.

Research methods

To solve the problem in this paper were used methods of analysis and synthesis, system analysis, comparison, logical generalization of results.

Scientific novelty

The scientific novelty of the work consists in the creation of the automated diagnostic system prototype based on deep learning algorithms for solving the classification problem of skin diseases, as well as approbation of the resulting prototype in practice using data provided by the international community for digital imaging of the skin ISDIS (13791 images of affected skin areas made with a dermoscope).

The practical value of the results

The results obtained can be used in future studies to improve the proposed prototype and classification model, taking into account the advantages and disadvantages of these results. Also, this prototype can be used to improve the performance of existing diagnostic systems.

Publications

1. Akymov V. Deep learning algorithms for classification of skin diseases / Vadym Akymov.// International Scientific Journal “Internauka”.–2018.–№9.

2. Akymov V. Алгоритми багаторівневого навчання для класифікації захворювань шкіри. / Vadym Akymov.// 20-th International Conference SAIT 2018 Kyiv, Ukraine, May 21-24, 2018. – 2018. – P. 121-122.

Keywords

Deep learning algorithms, convolutional neural networks (CNN), classification, skin diseases.

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

CNN – Convolutional Neural Network

ISDIS – The International Society for Digital Imaging of the Skin

ШМН – штучна нейронна мережа

Pooling — операція об’єднання в згорткових нейронних мережах

Batches — групи прикладів, що використовуються для навчання мережі

ВСТУП

Захворювання шкіри сьогодні належать до розповсюджених медичних проблем. Кількість таких захворювань постійно зростає, незважаючи на розвиток медичної галузі. Рак шкіри є поширеним злоякісним новоутворенням і займає друге рангове місце у структурі онкологічної захворюваності населення України. Такі хвороби діагностуються візуально, починаючи з клінічних обстежень, що можуть супроводжуватись дерматоскопічним аналізом, біопсією та гістопатологічною експертизою.

Особливий інтерес представляє автоматизована класифікація захворювань шкіри (як доброякісних, так і злоякісних) на базі зображення ураженої ділянки тіла. Глибокі згорткові нейронні мережі (Convolutional Neural Networks, CNN) показують потенціал для аналізу категорії дрібнозернистих зображень.

Навчальна вибірка даних представлена міжнародною спільнотою з цифрової обробки зображень шкіри ISDIS і містить 13791 зображень уражених ділянок шкіри, зроблених за допомогою дермоскопу.

1 ВИЗНАЧЕННЯ ЦІЛЕЙ ДОСЛІДЖЕННЯ

Завданням дипломного проекту є розробка моделі багаторівневого навчання для побудови прототипу автоматизованої діагностичної системи. Даний прототип служить для вирішення задачі класифікації шкірних захворювань на основі зображень враженої ділянки тіла людини.

2 НАВЧАЛЬНА ВИБІРКА ДАНИХ

Навчальна вибірка даних представлена міжнародною спільнотою з цифрової обробки зображень шкіри ISDIS (International Society for Digital Imaging of the Skin).

Академічний проект “Меланома” розроблений для полегшення аналізу цифрових зображень шкіри, його основна мета полягає у підтримці зусиль, спрямованих на зниження смертності від меланоми та непотрібних біопсій шляхом підвищення точності та ефективності раннього діагностування меланоми. Лікування не є проблематичним у випадку, коли дане

захворювання діагностується на ранніх стадіях. Цифрові зображення вражених шкірних ділянок використовуються з метою навчання професіоналів і громадськості в галузі розпізнавання злоякісних пухлин. В даний час відсутність стандартів в дерматології для зображення захворювань шкіри підриває якість і корисність наявних фотоматеріалів. ISIC розробляє запропоновані стандарти, технології та термінологію, що використовується при зображенні уражених ділянок шкіри. При цьому особлива увага приділяється питанням конфіденційності та сумісності (тобто можливість обміну зображеннями між різними клінічними платформами). Міжнародна програма з аналізу знімків шкіри ISIC розробила та поширює в загальному доступі архів (ISIC Archive) з великою кількістю зображень уражених ділянок шкіри. Цей архів служить загальнодоступним ресурсом для навчання, а також для розробки та тестування автоматизованих діагностичних систем.

Дерматологи, які спеціалізуються на раку шкіри, регулярно використовують загальну фотографію тіла та дерматоскопію як діагностичні засоби для виявлення меланоми. Загальна фотографія тіла дозволяє раннє виявлення активно змінюваних злоякісних уражень тіла. Дерматоскопи – це прості ручні прилади, які дозволяють уникнути відлисків поверхні та збільшують структури, що невидимі “неозброєним оком”. Спеціальні системи для збору, зберігання та вилучення зображень розроблені для лікарів з метою полегшення подальшого спостереження за допоміжними фотографіями та цифровим дермоскопічним моніторингом.

Вибірка містить як фото доброякісних утворень так і фото злоякісних пухлин: себорійний кератоз, плоскоклеточная карцинома меланома родимка, лентиго.

3 БАГАТОРІВНЕВЕ НАВЧАННЯ

Багаторівневі нейронні мережі в даний час стають одним з найпопулярніших підходів до створення систем штучного інтелекту, таких як розпізнавання мови, обробка природної мови, комп'ютерний зір і т.п., а також

популярним методом машинного навчання. Вони показують кращі результати в порівнянні з альтернативними методами в таких областях, як розпізнавання мови, обробка природної мови, комп'ютерний зір, медична інформатика та ін. Одна з причин успішного застосування багаторівневих нейронних мереж полягає в тому, що мережа автоматично виділяє з даних важливі ознаки, необхідні для вирішення завдання. В альтернативних алгоритмах машинного навчання ознаки повинні виділятися людьми, існує спеціалізований напрямок досліджень - інженерія ознак (feature engineering). Однак при обробці великих обсягів даних нейронна мережа справляється з виділенням ознак набагато краще, ніж людина.

Модель штучних нейронних мереж була запропонована в 1943 році, а сам термін глибоке навчання (deep learning) став широко використовуватися тільки починаючи з 2006 року. До цього застосовувалися терміни завантаження багаторівневих мереж (loading deep networks) і навчання багаторівневої пам'яті (learning deep memories).

Зростання популярності багаторівневих нейронних мереж, що відбувається в останні кілька років, можна пояснити трьома факторами. По-перше, відбулося істотне збільшення продуктивності комп'ютерів, в тому числі прискорювачів обчислень GPU (Graphics Processing Unit), що дозволило виконувати навчання мереж значно швидше і з більш високою точністю. Раніше наявних обчислювальних потужностей не вистачало для навчання скільки-небудь складної мережі, придатної для вирішення практичних завдань. По-друге, був накопичений великий обсяг даних, який необхідний для навчання багаторівневих нейронних мереж. По-третє, розроблені методи навчання нейронних мереж, що дозволяють швидко і якісно їх навчати, що складаються зі ста і більше шарів, що раніше було неможливо через проблеми зникаючого градієнта і перенавчання. Поєднання трьох чинників призвело до істотного прогресу в навчанні мереж і їх практичному використанні, що дозволило багаторівневим нейронним мережам зайняти лідируючу позицію серед методів машинного навчання [1].

3.1 Штучні нейронні мережі та їх складові

Штучні нейронні мережі були побудовані за принципом біологічних нейронних мереж, які представляють собою мережі нервових клітин, які виконують певні фізіологічні функції. Складовим елементом нейронних мереж є нейрони (представлені на рис. 3.1.1).

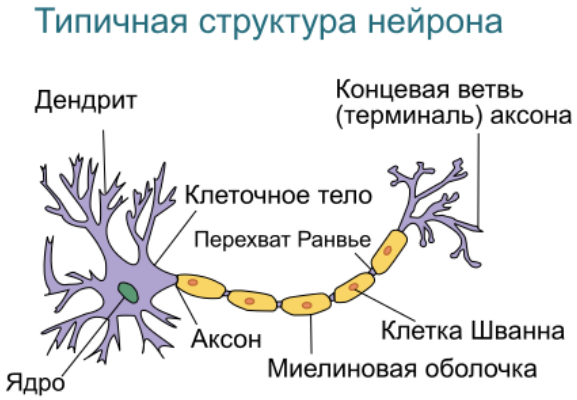

Рис.3.1.1 Типова структура нейрона

У нейрона є кілька функцій:

- Приймальня функція: синапси отримують інформацію;
- Інтегративна функція: на виході нейрона сигнал, який несе інформацію про суму сигналів в нейроні;
- Провідникова функція: по аксону проходить інформація до синапсів;
- Передаточна функція: імпульс, який досяг закінчення аксона, змушує медіатор передавати збудження наступному нейрону.

Синапсами називають зв'язку, по якій вихідні сигнали одних нейронів надходять на входи інших. Кожна зв'язка характеризується своєю вагою. Зв'язки з позитивним вагою називаються збудливими, а з негативним - гальмуючими. Вихід нейрона називається аксоном. У штучної нейронної мережі штучний нейрон – це деяка нелінійна функція, аргументом якої є лінійна комбінація всіх вхідних сигналів. Така функція називається активаційною. Потім результат активаційної функції посилається на вихід нейрона. Об'єднуючи такі нейрони з іншими, отримують штучну нейронну мережу [2].

3.2 Перцептрон

Елементарний перцептрон будується на основі сенсорних даних на вході – S-елементів, асоціативних елементів – A-елементів, і реагуючих елементів на виході – R-елементів. Набір S-елементів, пов'язаний з A-елементом, утворює асоціацію, і елемент активується після досягнення певного числа сигналів від S-елементів. A-елемент передає ваговий сигнал на суматорний R-елемент, і залежно від того, чи перевищує суму певної порога, R-елемент видає результат роботи перцептрона (рис.3.2.1).

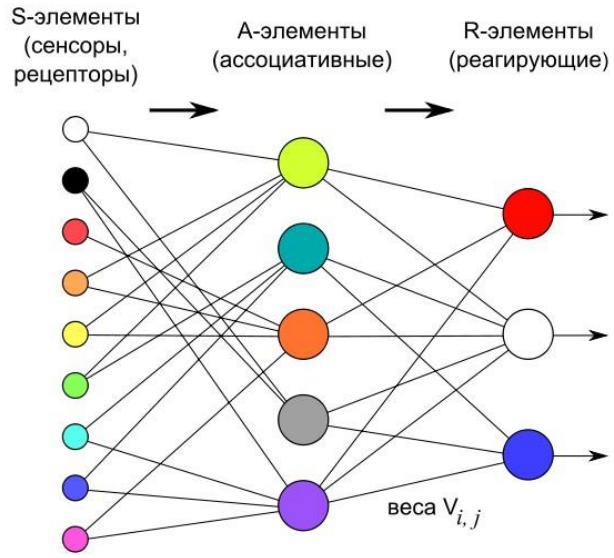

Рис.3.2.1 Перцептрон

Багатошаровий перцептрон будується з додатковими прихованими шарами A-елементів, розташованих між S-елементами і R-елементами. Принципіальна складність задач, що вирішуються багатошаровим перцептронним, є найвищою для класу перцептронів [3].

Навчання элементарного і багатошарового перцептрона полягає в зміні вагових коефіцієнтів зв'язків A - R. Перцептрон здатний працювати в режимі розпізнавання або узагальнення.

3.3 Багаторівневі нейронні мережі

Багаторівневими нейронними мережами називаються такі мережі, в яких є кілька прихованих шарів. Оскільки кожен прихований шар обчислює нелінійне перетворення попереднього шару, багаторівнева мережа може мати

значно більшу репрезентативну потужність (тобто може представляти значно складніші функції), ніж звичайна. При навчанні багаторівневої мережі важливо використовувати нелінійну функцію активації в кожному прихованому шарі. Це пов'язано з тим, що безліч шарів лінійних функцій самі вираховували б тільки лінійну функцію введення і, отже, не були б більш виразними, ніж при використанні тільки одного прихованого шару [4].

Головним достоїнством багаторівневих мереж є стисле представлення достатньо великої множини функцій. Можна показати, що існують функції, які k-шарова мережа може представити стисло, а (k-1) – шарова мережа не може цього зробити, якщо тільки вона не має експоненціально великої кількість елементів в прихованих шарах [5].

3.4 Доступність даних

За допомогою методу, описаного вище, можна покладатися тільки на марковані дані для навчання. Однак помічених даних часто буває недостатньо, і, отже, для багатьох завдань важко отримати достатню кількість прикладів для відповідності параметрам складної моделі. Наприклад, з огляду на високий ступінь виразності багаторівневих мереж, навчання при невеликій кількості даних призведе до перенавчання.

3.5 Локальний оптимум

Навчання малошарової мережі (з 1 прихованим шаром) з використанням контрольованого навчання зазвичай призводить до зближення параметрів з відповідними значеннями. Але при навчанні багаторівневої мережі, це працює набагато рідше. Зокрема, навчання нейронної мережі з використанням навчання з учителем включає в себе вирішення проблеми з невипуклою оптимізацією (наприклад, мінімізація помилки навчання $\sum_i \left\| h_W(x^{(i)}) - y^{(i)} \right\|^2$ в залежності від параметрів мережі W). У багаторівневій мережі з'являється велика кількість локальних оптимумів, тому навчання з градієнтним спуском перестає працювати [6].

3.6 Градієнтна дифузія

При використанні методу зворотного поширення помилки для обчислення похідних, градієнти, які поширюються від вихідного шару до більш ранніх шарів мережі, швидко зменшуються в міру збільшення глибини мережі. В результаті похідна від загальної вартості по відношенню до ваги в більш ранніх шарах дуже мала. Таким чином, при використанні градієнтного спуску ваги ранніх шарів повільно змінюються і більш ранні шари не можуть багато чому навчитися. Цю проблему часто називають "дифузією градієнтів" (diffusion of gradients).

3.7 Висновки

Багаторівневі нейронні мережі в даний час стають одним з найпопулярнішних підходів до створення систем штучного інтелекту, таких як розпізнавання мови, комп'ютерний зір, і т.п.. Штучні нейронні мережі були побудовані за принципом біологічних нейронних мереж, які представляють собою мережі нервових клітин, які виконують певні фізіологічні функції, а складовим елементом нейронних мереж є нейрони. Головним достоїнством багаторівневих мереж є стисле представлення достатньо великої множини функцій.

4 АРХІТЕКТУРА БАГАТОРІВНЕВОЇ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

Згорткові нейронні мережі (convolutional neural networks, CNN) – це широкий клас архітектур, основна ідея яких полягає в повторному використанні одних і тих же частин нейронної мережі для роботи з невеликими, локальними участками входів. Основним застосуванням даних мереж є обробка зображень. Ідея згорткових мереж багато у чому мотивована дослідженнями зорової кори головного мозку.

Розглянемо архітектуру згорткової мережі (рис 4.1), що була використана для класифікації захворювань. Дана мережа представляє собою послідовність шарів наступних типів: згортковий, агрегувальний (або шар підвибірки) та повнозв'язний [7].

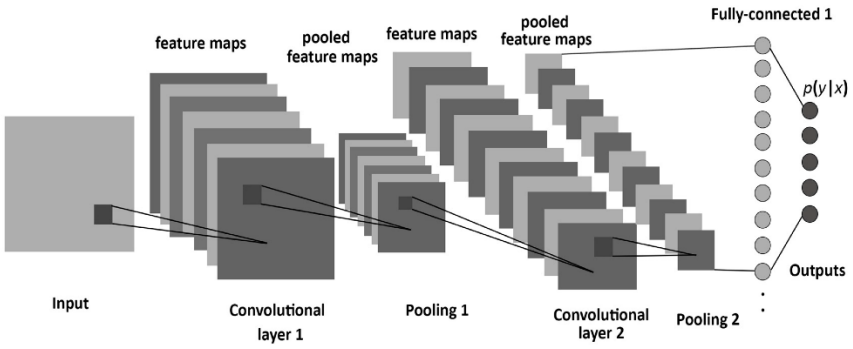

Рис.4.1 Архітектура багатошарової згорткової нейронної мережі

Вхідний шар. Містить сирі значення інтенсивностей пікселів. Зображення мають ширину і висоту 150px та три канали кольору R,G,B.

Шар згортки. Згортка – це лінійне перетворення вхідних даних. Нехай $x^l -$ карта ознак в шарі під номером l, тоді результат двовимірної згортки з ядром розміру $2d+1$ і матрицею ваги W розміром $(2d+1)$ x $(2d+1)$ на наступному шарі буде наступним:

$$y_{i,j}^l = \sum_{-d \le a,b \le d} W_{a,b} x_{i+a,j+b}^l,$$

де $y_{i,j}^l$ – результат згортки на рівні l, а $x_{i,j}^l$ – її вхід, тобто вихід всього попереднього шару. Інакше кажучи, щоб отримати компоненту (i,j) наступного рівня, необхідно використати лінійне перетворення до квадратного вікна попереднього рівня, тобто скалярно помножити пікселі із даного вікна на ядро згортки (рис 4.2).

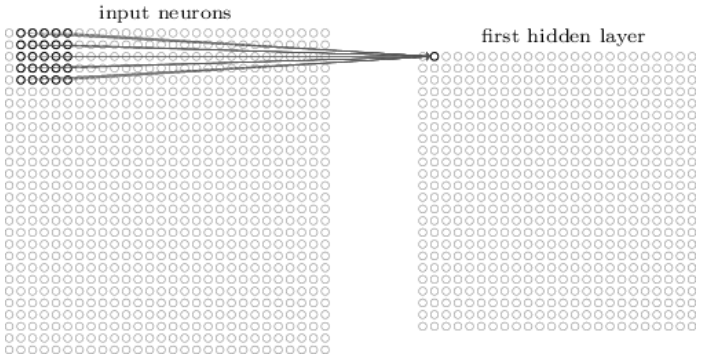

Рис.4.2 Візуалізація операції згортки

Для внесення нелінійності в модель використовується функція активації випрямляч (ReLU): $h(x) = \max(0, x)$.

Агрегувальний шар. Використовується для зменшення розмірності вхідної карти ознак за рахунок їх узагальнення і втрати незначної частини інформації про їх положення. В рамках даної задачі рекомендовано використовувати метод вибору максимального елемента:

$$x_{i,j}^{l+1} = \max_{-d \le a \le d, -d \le b \le d} z_{i+a,j+b}^{l},$$

де d – розмір вікна шару підвибірки.

Повнозв'язний шар. Після декількох проходжень згортки зображення і шарів підвибірки система перебудовується від конкретної сітки пікселів з високим розширенням до більш абстрактних карт ознак. Як правило на кожному наступному шарі збільшується кількість каналів і зменшується розмірність зображення в рамках кожного каналу. Ці дані об'єднуються і передаються на звичайну повнозв'язну нейронну мережу, яка представлена наступним чином (рис 4.3):

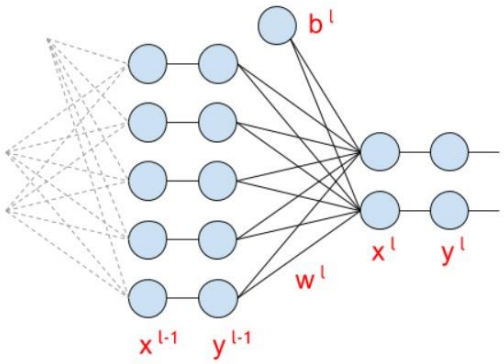

Рис.4.3 Візуалізація повнозв'язного шару

$$x_i^l = \sum_{k=0}^{m} w_{ki}^l y_k^{l-1} + b_i^l, \forall\, i \in (0, \dots, n)$$

В якості функції активації для кінцевого шару використовується SoftMax, що перетворює вхідний вектор в вектор ймовірностей приналежності об'єкта до відповідного класу [8].

5 ФУНКЦІЇ АКТИВАЦІЇ НЕЙРОНІВ

Для реалізації нелінійності при активації нейрона, його активність, крім різних видів суматорів і систем ваг на входах, визначається функцією одного

аргументу – функцією активації. Нейрон в цілому реалізує скалярну функцію векторного аргументу, а вихідний сигнал нейрона визначається видом функції активації і може бути дійсним або цілим. Функція активації застосовується до зваженої суми сигналів на вході нейрона. Таким чином, активність нейрона повністю визначається його параметрами – вагами і його функцією активації. Існує безліч передавальних функцій, що застосовуються на практиці використання нейронних мереж, деякі з них служать для реалізації нелінійності системи. Вибір тієї чи іншої функції активації часто залежить від умов завдання і структури мережі. Деякі з розглянутих передавальних функцій застосовуються тільки в застарілих системах або в цілях навчання, але вважаються класичними і згадуються щоразу при вивченні ШНМ [9].

5.1 Порогова функція активації

Порогова функція Хевісайда є найбільш простою кусочно-лінійною передавальною функцією. Ця функція використовувалася в класичному перцептроні, тобто наразі використовується в основному з метою навчання теорії нейронних мереж. Функція Хевісайда – кусочно-постійна функція, яка рівна нулю для від’ємних значень аргумента і одиниці для невід’ємних значень. В нулі дана функції не визначена, але дуже часто її визначають в цій точці деяким числом щоб область визначення функції містила всі точки дійсної осі. Частіше за все взагалі неважливо якого значення набуває дана функція в нулі.

$$\theta(x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0. \end{cases}$$

5.2 Лінійна функція активації

При використанні нескладної кусочно-лінійної передавальної функції сигнал на виході нейрона лінійно пов'язаний зі зваженою сумою сигналів на його вході. В даний момент лінійна функція на практиці також використовується дуже рідко.

5.3 Сигмоїдальна функція активації

Сигмоїд є монотонно зростаючою всюди диференційованою S-подібною нелінійною функцією з насиченням. Забезпечує посилення слабких сигналів і запобігає насичення сильних сигналів. Є однією з найбільш поширених передавальних функцій, часто використовується в нейронних мережах донині. Введення сигмоїдальних функцій було зумовлено недостатньою гнучкістю класифікаторів на основі порогових передавальних функцій і дозволило перейти від жорсткої однорозрядної логіки до більш гнучкої поведінки і адаптивної параметризації нейронних мереж. Найбільш часто використовуваним прикладом сигмоидальної функції є логістична передавальна функція.

$$OUT = \frac{1}{1 + \exp(-\alpha Y)},$$

де α – параметр нахилу сигмоїдальної функції активації. Можна побудувати функції різної степені крутизни шляхом зміни даного параметру.

5.4 Функція активації гіперболічний тангенс

Ще одним прикладом сигмоїдальної функції активації є гіперболічний тангенс, який задається наступним виразом:

$$OUT = th\left(\frac{Y}{a}\right)$$

де α – це також параметр, який впливає на нахил сигмоїдальної функції. Функція гіперболічного тангенса відрізняється від розглянутої вище логістичної кривої тим, що її область значень лежить в інтервалі (-1; 1), що в деяких випадках може спростити завдання навчання ШНМ.

5.5 Функція активації ReLU (Rectified Linear Unit)

У неглибоких ШНМ використовуються нелінійні функції активації. Часто зустрічаються різновиди сигмоїдальних і тангенціальних передавальних функцій, які є нелінійними, але на практиці навчання глибоких ШНМ такі функції можуть привести до проблем з загасанням або збільшенням градієнтів. Функція ReLU є випрямленою лінійною функцією і на даний момент вважається більш простим і ефективним з точки зору

обчислювальної складності варіантом передавальної функції. Похідна ReLU дорівнює або 0, або 1, від чого її застосування запобігає розростання і загасання градієнтів, і призводить до проріджування ваг, що позитивно позначається на обчислювальної здатності ШНМ. Передавальна функція ReLU є одним з останніх успіхів в області методів налаштування глибоких нейронних мереж [10].

$$f(x) = \max(0, x)$$

де x – вхід нейрона.

Сьогодні існує сімейство різних модифікацій ReLU, які вирішують проблеми надійності цієї передавальної функції при проходженні через нейрон великих градієнтів: Leaky ReLU, Parametric ReLU, Randomized ReLU.

5.6 Висновки

В даному розділі описано основні функції активації, які використовуються в архітектурі нейронних мереж для реалізації нелінійності при активації нейрона. Функція активації застосовується до зваженої суми сигналів на вході нейрона. Таким чином, активність нейрона повністю визначається його параметрами – вагами і його функцією активації. Вибір тієї чи іншої функції активації часто залежить від умов завдання і структури мережі. Деякі з розглянутих передавальних функцій застосовуються тільки в застарілих системах або в цілях навчання.

В рамках розробки прототипу автоматизованої діагностичної системи для класифікації шкірних захворювань використовуватиметься функції активації ReLu для внесення нелінійності на виході конволюційних шарів згорткової нейронної мережі.

6 ПРОГРАМНІ ІНСТРУМЕНТИ ПОБУДОВИ НЕЙРОННИХ МЕРЕЖ

В останні роки навчання нейронних мереж перетворилось в більшій мірі в інженерну дисципліну. З’явилось багато зручних бібліотек, які дозволяють побудувати модель нейронної мережі (будь-якої глибини), сформувати для неї граф обчислень, автоматично порахувати градієнти і виконати процес навчання. Причому зробити це можна як на процесорі, так і

на відеокарті, що значно прискорює виконання. Є і бібліотеки загального призначення, які дозволяють створити будь-який граф обчислень і містять готові імплементації компонент нейронних мереж: звичайні слої, згорткові, рекурентні, сучасні алгоритми оптимізації.

Сьогодні існує більше десятка бібліотек багаторівневого навчання, деяким з яких притаманна вузько направлена специфіка. Грунтуючись на рекомендаційної інформації з широкого кола джерел, для подальшого розгляду було вибрано такі бібліотеки: Theano, Torch, Caffe, Neon і Keras [11]. Базуючись на двох простих критеріях, таких як виразність і розмір активної спільноти розробників, виділимо 3 основні бібліотеки: Tensorflow, Theano і Torch.

Довгий час лідером була бібліотека Theano, розроблена в університеті Монреаля. Мінусом даного фреймворку є наявність додаткової фази компіляції графу обчислень, що потребує велику кількість часу для налаштування будь-якої архітектури нейронної мережі. В 2015 році Google випустила (з відкритим вихідним кодом) бібліотеку Tensorflow. Вона має більш виразний інтерфейс і, на відміну від Theano, дана бібліотека була розроблена для промислового використання, а не лише для науково-дослідницьких задач. Саме тому вона містить такі важливі функції, як можливість запуску в мобільному середовищі, можливість запуску на кластері з CPU та GPU та масштабування даного обчислювального кластеру.

Існують два типи бібліотек: символьні і імперативні. У символьних бібліотеках набагато більше можливостей багаторазового використання пам'яті, а оптимізація на основі графів залежностей здійснюється автоматично. Найпопулярнішими символьними (symbolic) бібліотеками на даний час є TensorFlow і Theano. На відміну від Theano, Tensorflow не орієнтований тільки на навчання нейронних мереж, тому можна використовувати колекції графів і черги в якості складових частин для високорівневих компонентів. Якщо необхідно навчати масштабні моделі і використовувати багато зовнішньої пам'яті, то Theano буде дуже повільно

працювати через необхідність компіляції коду C / CUDA в бінарний код. TensorFlow має прозору модульну архітектуру з користувацьким інтерфейсом для кожного з них.

Архітектура Theano досить непроста: весь код - це Python, де код C / CUDA упакований як рядок Python. В такому коді важко орієнтуватися, його непросто відлагоджувати і проводити рефакторинг. Більш того, візуалізація графів в TensorFlow реалізована значно ефективніше, ніж в Theano.

Основним недоліком бібліотеки Torch є те, що вона написана на Lua – скриптовій мові програмування, яка не набула масової поширеності на ринку. Відсутність необхідності вивчати нову мову програмування дає можливість залучення великої аудиторії Python розробників, тому вибір здійснюється між Tensorflow та Theano.

6.1 Огляд бібліотеки Torch для багаторівневого навчання

Бібліотека Torch розроблена як бібліотека для обчислень в наукових цілях і підтримує безліч технологій. Бібліотека дозволяє гнучко працювати з нейронними мережами на досить низькому рівні. Для реалізації нейронної мережі в Torch необхідно написати власний цикл навчання, в якому оголошується функція замикання, що обчислює відповідь мережі. Це замикання передається в функцію градієнтного спуску для поновлення ваг мережі. Використання Torch не вимагає серйозних витрат за часом написання програмного коду, крім того, мережі Torch мають перевагу порівняно з мережами інших фреймворків швидкістю класифікації і наявністю вичерпної документації.

6.2 Огляд бібліотеки Theano для багаторівневого навчання

Theano існує в першу чергу як розширення мови Python, що дозволяє ефективно обчислювати математичні вирази, що містять багатовимірні масиви. У бібліотеці реалізований базовий набір інструментів для побудови ШНМ. Процес створення моделі і визначення її параметрів вимагає написання об'ємного коду, що включає реалізацію класу моделі, самостійного визначення її параметрів, реалізацію методів, що визначають

функцію помилки, правило обчислення градієнтів, спосіб зміни ваг. Theano є найбільш гнучким фреймворком для побудови ШНМ.

6.3 Огляд бібліотеки Caffe для багаторівневого навчання

Caffe реалізована на C ++ і має обгортки для Python і Matlab. Топологія ІНС, вихідні дані і спосіб навчання задаються за допомогою конфігураційних protobuf-файлів (технологія і протокол сериализации даних, що перевершує по ефективності xml і json) в форматі prototxt. Побудова структури мережі виконується з простотою, зручністю і наочністю. Із запропонованих бібліотек є найбільш зручною у використанні, крім того, перевершує інші розглядаються фреймворки в швидкості навчання. Бібліотека Caffe підтримується досить великим співтовариством розробників і користувачів і на сьогоднішній день є найпоширенішою бібліотекою глибокого навчання широкого кола застосування.

6.4 Огляд бібліотеки Tensorflow для багаторівневого навчання

TensorFlow - це бібліотека програмного забезпечення з відкритим вихідним кодом для задач машинного навчання, розроблена Google. Вона дозволяє створювати і навчати нейронні мережі різної архітектури для виявлення і розпізнавання зразків і пошуку взаємозв'язків. TensorFlow також включає в себе TensorBoard, який являє собою засіб візуалізації в браузері для оцінки ефективності навчання і мережевих параметрів моделі [12].

TensorFlow досягає своєї продуктивності завдяки паралелізації завдань між центральним і графічними процесорами. Ядро кожної операції реалізовано на C++ з використанням бібліотек Eigen і cuDNN для кращої продуктивності.

Кожне обчислення в TensorFlow представляється як граф потоку даних, він же граф обчислень. Граф обчислень є моделлю, яка описує як будуть виконуватися обчислення. Важливо зауважити, що побудова графа обчислень і виконання операцій в заданій структурі - два різних процеси. Граф складається з плейсхолдерів (tf.Placeholder), змінних (tf.Variable) і операцій.

У ньому відбувається обчислення тензорів - багатовимірних масивів, які втім можуть бути числом або вектором.

Графи виконуються в сесіях (tf.Session). Існують два типи сесій - звичайні і інтерактивні (tf.InteractiveSession); інтерактивна сесія підходить для виконання в консолі. Сесія зберігає стан змінних (Variables) і черг (queues). Явне створення сесій і графів гарантує належне звільнення ресурсів пам'яті [14].

У графі кожна вершина має 0 або більше входів і 0 або більше виходів, і являє собою реалізацію операції. Тензори представлені ребрами графа, а саме масивами довільного розміру (тип масиву вказується під час побудови графа). Особливі вершини, управляючі залежності (control dependencies), також можуть бути в графі: вони вказують, що вихідний вузол для контрольної залежності повинен закінчити виконання до того, як вузол одержувача контрольної залежності почне виконуватися.

Кожна операція має назву і являє собою абстрактне обчислення (наприклад, підсумовування). У операції можуть бути атрибути: наприклад, можливість зробити операцію поліморфною для різних типів тензорів. Ядро - специфічна реалізація операції, яка може виконуватися на певному типі пристрою (центральний або графічний процесор).

Змінна – особливий вид операції, який повертає вказівник на постійно мінливий тензор: така змінна не зникає після одиничного використання графа. Вказівники на подібні тензори передаються численним операціями, які потім змінюють вказаний тензор. У завданнях машинного навчання, параметри моделі зазвичай зберігають тензори в змінних, які оновлюються на кожному кроці навчання.

Дана робота виконана на єдиному пристрої з використанням CPU.

6.5 Висновки

В даному розділі було проаналізовано основні переваги та недоліки таких програмних інструментів для практичної побудови та навчання нейронних мереж, як: Theano, Torch, Caffe, Neon і Keras, Tensorflow.

В рамках даної роботи було обрано для імплементації прототипу програмну платформу Tensorflow, в якому кожна модель представлена у вигляді графу потоку данних (або графу обчислень). Серед основних його переваг є висока степінь паралелізації обчислень графу, можливість виконання обчислювальних операцій як на центральному процесорі, так і на потужних дискретних відеокартах на базі програмної платформи CUDA. Також Tensorflow має досить велику аудиторію користувачів у вигляді Python розробників та багатий пакет супроводжувальної документації. В перелік стандартної поставки входять допоміжні інструменти розробки, такі як Tensorboard для візуалізації графу обчислень та результатів навчання.

Також в реєстрі Docker Hub є готові образи, що містять всі пакети та їх залежності для імплементації сервісів з підтримкою програмної платформи Tensorflow.

7 НАВЧАННЯ КЛАСИФІКАЦІЙНОЇ МОДЕЛІ

7.1 Алгоритм градієнтного спуску та його модифікації

Градієнтний спуск – це ітераційний алгоритм оптимізації першого порядку, в якому для знаходження локального мінімуму функції здійснюються кроки, пропорційні протилежному значенню градієнту (або наближеного градієнту) функції в поточній точці. Якщо натомість здійснюються кроки пропорційно самому значенню градієнту, то відбувається наближення до локального максимуму цієї функції [9].

Відомо, що напрямок найбільшого зростання будь-якої функції, наприклад $F = f(x_1, x_2, \dots, x_n)$ характеризується її градієнтом:

$$gradf = \frac{\partial f}{\partial x_1} e_1 + \frac{\partial f}{\partial x_2} e_2 + \cdots + \frac{\partial f}{\partial x_n} e_n ,$$

де $e_1, e_2, \dots, e_n$ – одиничні вектори у напрямку координатних осей. Процес відшукання точки мінімуму функції F за методом градієнтного спуску полягає в наступному: на початку вибираємо деяку початкову точку $M_0(x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})$ і обчислюємо в ній градієнт функції F. Далі, робимо крок у антиградієнтному напрямку $x^1 = x^0 - a^1 gradf(M_0)$ (де $a^1 > 0$). У

результаті отримуємо нову точку $M_1(x_1^{(1)}, x_2^{(1)}, \dots, x_n^{(1)})$, значення функції в якій зазвичай менше за значення функції в точці M_0. Якщо ця умова не виконується, тобто значення функції не змінилось або навіть зросла, то потрібно зменшити крок $a^1(a^2 = a^1/2)$, після чого, у новій точці обчислюємо градієнт і знову робимо крок у зворотному до нього напрямку $x^2 = x^1 - a^2 gradf(M_1)$.

Процес продовжується до отримання найменшого значення цільової функції. Строго кажучи, момент закінчення пошуку настане тоді, коли рух з отриманої точки, при виборі будь-якого кроку, призводить до зростання значення цільової функції. Якщо мінімум функції досягається всередині розглядуваної області, то в цій точці градієнт дорівнює нулю, що також може служити сигналом про закінчення процесу оптимізації [10].

Формули для частинних похідних можна отримати в явному вигляді лише в тому випадку, коли цільова функція задана аналітично. В іншому випадку ці похідні обчислюються за допомогою чисельного диференціювання:

$$\frac{\partial f}{\partial x_i} \approx \frac{f(x_1, \dots, x_i + \Delta x_i, \dots, x_n) - f(x_1, \dots, x_i, \dots, x_n)}{\Delta x_i}, i = 1,2, \dots, n$$

Необхідно зауважити, що знайти точку мінімуму функції F можна також шляхом зведення багатовимірної задачі оптимізації до послідовності одновимірних задач на кожному кроці оптимізації. Такий спосіб називається методом покоординатного спуску. Різниця полягає в тому, що в методі градієнтного спуску напрямок оптимізації визначається градієнтом цільової функції, тоді як у методі покоординатного спуску проводиться спуск на кожному кроці вздовж одного з координатних напрямків [11].

В машинному навчанні градієнтний спуск розглядається як спосіб підбору векторів синаптичних ваг w. Розглянемо випадок лінійного класифікатора. Нехай $y^*: X \to Y$ – цільова залежність, яка відома лише на об'єктах навчальної вибірки: $X^l = (x_i, y_i)_{i=1}^l, y_i = y^*(x_i)$. Знайдемо алгоритм $a(x, w)$, що апроксимує залежність y^*. У випадку лінійного класифікатора

шуканий алгоритм має вигляд: $a(x,w) = \varphi\left(\sum_{j=1}^{n} w_j x^j - w_0\right)$, де $\varphi(z)$ грає роль функції активації (в найпростішому випадку можна закласти $\varphi(z) = sign(z)$). Згідно з принципом мінімізації емпіричного ризику, для цього достатньо вирішити оптимізаційну задачу: $Q(w) = \sum_{i=1}^{l} L(a(x_i,w), y_i) \rightarrow min_w$, де L(a, y) – задана функція втрат. Для мінімізації якраз і необхідно використати алгоритм градієнтного спуску: $w := w - \eta\nabla Q(w)$, де η – додатній параметр, який називається темпом навчання (learning rate).

Існує 2 підходи в реалізації градієнтного спуску:

- Пакетний (batch), коли на кожній ітерації навчальна вибірка переглядається частково (або взагалі цілком), тільки після чого змінюється w. Це потребує великих обчислювальних затрат.
- Стохастичний, коли на кожній ітерації алгоритму з навчальної вибірки випадковим чином обирається лише один об’єкт. Таким чином вектор w налаштовується кожен раз на новобраний об’єкт [12].

7.2 Метод зворотного поширення помилки для мінімізації помилки

Це ітеративний градієнтний метод навчання багатошарового перцептрону, який використовується з метою мінімізації помилки його роботи. Основна ідея цього методу полягає в поширенні сигналів помилки від виходів мережі до її входів, в напрямку, зворотному прямому поширенню сигналів у звичайному режимі роботи. Для можливості застосування методу зворотного поширення помилки функція активації нейронів повинна бути диференційованою [13].

Завчасно невідомо які ваги повинні мати з’єднання схованого шару нейронної мережі, проте можливо визначити наскільки швидко змінюється помилка при зміні ваги кожного індивідуального з’єднання. Взагалі, необхідно знайти шлях найкрутішого спуску. Єдина складність в тому, що необхідно працювати в багатовимірному просторі. Почнемо з обчислення похідних функції помилок по відношенню до кожного окремого об’єкта тренувального набору. Кожен нейрон прихованого шару може впливати на велику кількість вихідних нейронів. Тому нашою стратегією буде динамічне

програмування. Як тільки ми маємо похідні функції помилки для одного прихованого шару, ми використаємо їх для обчислення похідних нижнього шару. Введемо деякі позначення (рис.7.2.1):

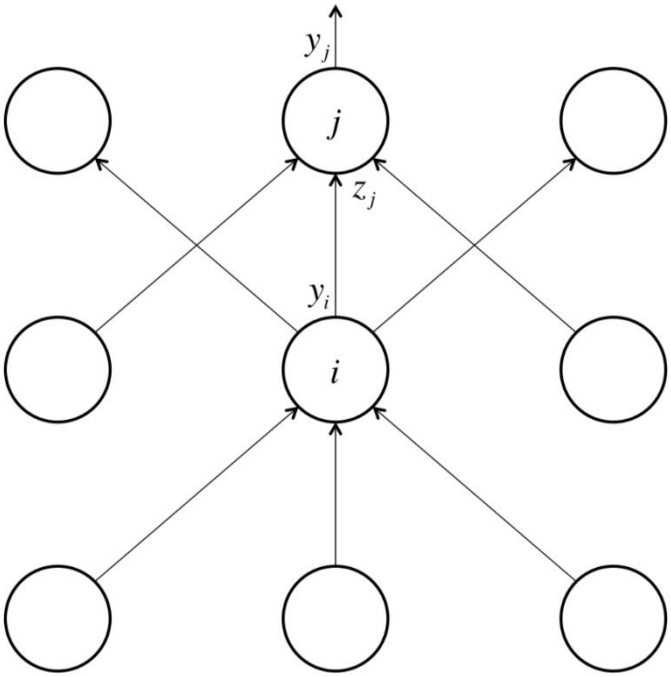

Рис.7.2.1. Діаграма для виведення алгоритму зворотного розповсюдження помилки

Визначимо похідні функції помилки для вихідного шару мережі:

$$E = \frac{1}{2}\sum_{j \in output}(t_j - y_j)^2 => \frac{\partial E}{\partial y_j} = -(t_j - y_j)$$

Розглянемо тепер індуктивний крок. Припустимо, що відомі похідні для j-го шару. Задача в тому, щоб порахувати похідні для шару нижче, і-го. Для цього необхідно зібрати інформацію про те як вихід нейрону з і-го шару впливає на кожний нейрон j-го шару [14].

$$\frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial z_j}\frac{dz_j}{dy_i} = \sum_j w_{ij}\frac{\partial E}{\partial z_j}$$

Бачимо, що:

$$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial z_j} = y_j(1 - y_j)\frac{\partial E}{\partial y_j}$$

Поєднуючи ці два твердження, виразимо похідні і-го шару в термінах похідних j-го шару:

$$\frac{\partial E}{\partial y_i} = \sum_j w_{ij} y_j (1 - y_j) \frac{\partial E}{\partial y_j}$$

Виконавши всі кроки алгоритму динамічного програмування, заповнивши таблицю всіх часткових похідних (функції помилки відносно виходів кожного слою), можна визначити як змінюється функція помилки залежно від вагів. Це дозволяє зрозуміти як модифікувати ваги після кожного об'єкта тренувального набору [15].

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial z_j}{\partial w_{ij}} \frac{\partial E}{\partial z_j} = y_i y_j (1 - y_j) \frac{\partial E}{\partial y_j}$$

В результаті ваги з'єднань будуть модифікуватися наступним чином:

$$\Delta w_{ij} = -\sum_{k \in dataset} \in y_i^{(k)} y_j^{(k)} (1 - y_j^{(k)}) \frac{\partial E^{(k)}}{\partial y_j^{(k)}}$$

7.3 Висновки

В рамках даного розділу були розглянуті математичні основи методів навчання нейронних мереж, зокрема ітераційний алгоритм оптимізації першого порядку – градієнтний спуск, в якому для знаходження локального мінімуму функції здійснюються кроки, пропорційні протилежному значенню градієнту. Враховуючи проблеми класичного алгоритму градієнтного спуску з локальними мінімумами, було розглянуто основні модифікації даного алгоритму, такі як порційний та стохастичний.

Для навчання багатошарового перцептрону використовується метод зворотного поширення помилки, який являє собою ітеративний градієнтний метод і його ідея полягає в поширенні сигналів помилки від виходів мережі до її входів, в напрямку, зворотному прямому поширенню сигналів у звичайному режимі роботи.

8 ПРОБЛЕМИ НАВЧАННЯ БАГАТОРІВНЕВИХ НЕЙРОННИХ МЕРЕЖ ТА ЇХ ВИРІШЕННЯ

8.1 Затухаючий градієнт

Проблема зникаючого градієнта – це складність, яка виникає при навчанні штучних нейронних мереж з використанням методів навчання на

основі градієнта і зворотного поширення помилки. В таких методах будь-яка вага нейронної мережі оновлюється пропорційно градієнту функції помилки щодо поточної ваги на кожній ітерації навчання. Стандартні функції активації, такі як гіперболічний тангенс, мають градієнти в діапазоні (-1, 1), а метод зворотного поширення помилки обчислює їх по ланцюговому правилом. Після множення цих чисел для обчислення градієнтів "фронтальних" шарів в n-шарової мережі, що означає, що градієнт (сигнал помилки) експоненціально зменшується разом з n, а передні шари навчаються дуже повільно.

Коли використовуються функції активації, похідні яких можуть брати великі значення, є ризик зіткнутися з exploding gradient problem. Можливими рішеннями є:

- Багаторівнева ієрархія: шар мережі попередньо навчається, використовуючи методи навчання без учителя, а потім його значення регулюється за допомогою методу зворотного поширення помилки. Таким чином кожен шар мережі вивчає стисле уявлення спостережень, яке подається на наступний шар [16];
- Довга короткострокова пам'ять: різновид архітектури рекурентних нейронних мереж. Коли величини помилки розповсюджуються в зворотному напрямку від вихідного шару, помилка не випускається з пам'яті LSTM-блоку. Вона безперервно передається назад кожному з вентилів, поки вони не будуть навчені відкидати подібні значення [17];
- Залишкові мережі (Residual networks): один з найбільш ефективних методів вирішення проблеми зникаючого градієнта є використання залишкових нейронних мереж (ResNets). Більш глибока мережа матиме вищу помилку навчання, ніж малошарова мережа. Команда Microsoft Research виявила, що поділ глибокої мережі на частини (скажімо, кожна частина являє собою три шари мережі) і передача вхідних даних в кожен фрагмент до наступного фрагмента (поряд із залишковим виходом З шматка мінус вхідні дані знову введеного фрагмента)

допомогли усунути більшу частину цієї проблеми зі зникненням градієнта. Ніяких додаткових параметрів або змін в алгоритмі навчання не потрібно. ResNets показали нижчу помилку навчання (і тестову помилку), ніж їх більш малошарові аналоги, ввівши його ще раз виходів з більш дрібних верств в мережі для компенсації зникаючих даних.

8.2 Сигмоїдальні активаційні функції

Використання сигмоїдальних активаційних функцій може викликати проблеми в навчанні багаторівневих мереж, а саме значення активацій в кінцевому шарі будуть близькі до нуля на ранніх етапах навчання, сповільнюючи цей процес. Були запропоновані альтернативні активаційні функції, які не так страждають від обмеження. Вибір відповідних ваг і momentum schedule в імпульсному стохастичною градієнтному спуску (momentum-based stochastic gradient descent) значно впливають на здатність навчати багаторівневі мережі.

8.3 Алгоритми регуляризації для уникнення проблеми перенавчання

- L1-регуляризація: відбувається зміна нерегуляризованої цільової функції шляхом додавання суми абсолютних значень ваг:

$$J = J_0 + \frac{\lambda}{n}\sum_{W}|W|$$

  При використанні L1-регуляризації відбувається наближення одного або більше вагових значень до 0,0, тому відповідна функція (функція) більше не потрібна [18]. Цей ефект називається селекцією функцій (вибір функції);

- L2-регуляризація (також відома як розпад ваги). На відміну від L1-регуляризації, в L2 вага зменшуються за величиною, пропорційну вагам:

$$J = J_0 + \frac{\lambda}{2n}\sum_{W}W^2$$

- Dropout не впливає на значення цільової функції: змінюється структура мережі. Кожний нейрон видаляється з мережі з деякою імовірністю p. По отриманій розрядженій мережі робиться зворотне розповсюдження помилок, для залишкових ваг виконується градієнтний крок. Після цього видалені нейрони відновлюються в мережі. При навчанні нейромережі вихід кожного нейрона домножується на (1-p). Так буде отримано математичне очікування відповіді мережі по її 2N (де N - кількість нейронів в мережі) архітектурам. Навчена таким чином мережа є результатом усереднення 2N мереж. Окрема нейронна мережа, навчена за допомогою ранньої зупинки, має занадто велику помилку, однак, середнє кількох нейронних мереж призводить до істотного зниження помилок [19].
- Штучне розширення даних для навчання.

8.4 Висновки

В рамках даного розділу були висвітлені основні проблеми навчання багаторівневих нейронних мереж та методи їх вирішення. Серед проблем виділено наступні:

- Затухаючий градієнт – складність, що виникає при навчанні штучних нейронних мереж з використанням градієнтних методів. Після цілого ланцюга обчислень значення градієнту експоненціально зменшується в напрямку до входів мережі. Для вирішення даної проблеми необхідно правильно обирати функції активації та використовувати правильну преініціалізацію параметрів мережі.
- Проблема перенавчання мережі, в результаті якої нейронна мережа втрачає можливість прогнозування нових значень. Для вирішення даної проблеми використовуються техніки L1 та L2 регуляризації, також техніка Dropout.
- Проблеми з сигмоїдальними активаційними функціями, при використанні яких значення активацій в кінцевому шарі мережі будуть близькі до нуля на ранніх етапах навчання. Для вирішення даної

проблеми необхідно використовувати альтернативні активаційні функції.

9 ІМПЛЕМЕНТАЦІЯ ПРОТОТИПУ СИСТЕМИ КЛАСИФІКАЦІЇ ШКІРНИХ ЗАХВОРЮВАНЬ

9.1 Реалізація графу обчислень багаторівневої класифікаційної моделі

Для імплементації класифікаційної моделі у вигляді згорткової нейронної мережі було використано бібліотеку машинного навчання і багаторівневого дослідження нейронних мереж від компанії Google, в рамках науково-дослідницької організації Machine Intelligence – некомерційної організації, основною ціллю якої створення безпечного штучного інтелекту, а також вивчення потенційних загроз та можливостей, які можуть з’явитися при створенні штучного інтелекту. Дана програмна платформа гнучка і масштабована, може бути використана як на смартфонах, так і на потужних комп’ютерах. Так, наприклад, дана бібліотека використовується в багатьох застосуваннях: Google Photos, Google Translate, автовідповідач Inbox і других застосунках.

В основі Tensorflow – графи потоків даних. Дана бібліотека не потребує написання організації алгоритмів, достатньо описати архітектуру нейронної мережі та її параметри. Далі все вказані данні відправляються в сесію Tensorflow, де відбуваються всі обчислення, які необхідні для отримання результатів [20].

Обчислення представляються у вигляді направленого графа, шляхи по яким дані переміщаються – ребра графа. Дані представлені у вигляді тензорів – багатовимірних масивів даних зі змінним розміром.

Побудуємо архітектуру згорткової нейронної мережі. Модуль шарів TensorFlow забезпечує API високого рівня, що полегшує побудову нейронної мережі. Він надає методи, які полегшують створення щільних (повністю з'єднаних) шарів і згорткових шарів, додавання функцій активації та застосування регуляризації dropout. Згорткові нейронні мережі (CNN) застосовують серію фільтрів до необроблених піксельних даних зображення,

щоб витягувати і вивчати особливості вищого рівня, які модель потім може використовувати для класифікації. CNN містить три компоненти:

- Конволюційні шари, які застосовують певну кількість фільтрів згортки до зображення. Для кожного субрегіону шар виконує набір математичних операцій для одержання одного значення на карті функцій виходу. Конволюційні шари зазвичай застосовують функцію активації ReLU до виводу для введення нелінійності в модель.
- Агрегувальний шари, які допомагають знімати дані зображень, витягнуті згортковими шарами, щоб зменшити розмірність карти об'єкта, щоб зменшити час обробки. Загальнопоширений алгоритм злиття - це макс-об'єднання, яке витягує субрегіони карти об'єктів (наприклад, 2х-піксельні плитки), зберігає їх максимальне значення та відкидає всі інші значення.
- Щільні (повністю з'єднані) шари, які виконують класифікацію за ознаками, отриманими конволюційними шарами, і зменшуються за допомогою об'єднуючих шарів. У щільному шарі кожен вузол у шарі з'єднаний з кожним вузлом у попередньому шарі.

Як правило, згорткова нейронна мережа складається зі стеків згорткових модулів, які виконують функції вилучення. Кожен модуль складається з згорткового шару, за яким іде агрегувальний шар. На останньому модулі згортки розміщується один або кілька повнозв’язних шарів, які виконують класифікацію. Остаточний повнозв’язний шар в CNN містить один вузол для кожного цільового класу в моделі (всі можливі класи, які може передбачити модель), з функцією активації softmax для створення значення між 0-1 для кожного вузла (сума усіх значень softmax шару дорівнює 1). Ми можемо інтерпретувати значення softmax для даного зображення як відносні вимірювання того, наскільки імовірним є те, що зображення входить до кожного цільового класу.

Конволюційний шар №1: застосовується 32 фільтрів 5x5x3 (витягання 5x5-піксельних субрегіонів), з функцією активації ReLU. У першому

згортковому шарі необхідно застосувати до вхідного шару 32 5x5x3 фільтрів з функцією активації ReLU. Ми можемо використовувати метод conv2d() у модулі шарів, щоб створити цей шар наступним чином:

```
# [Convolution #1]
with tf.variable_scope('conv1') as scope:
    filters_count = 32
    kernel_size = 5

    conv1 = tf.layers.conv2d(
        inputs=images,
        filters=filters_count,
        kernel_size=kernel_size,
        activation=tf.nn.relu,
        kernel_initializer=tf.random_normal_initializer(),
        padding='SAME')
```

Агрегувальний шар №1: виконує максимальне об'єднання з фільтром 2x2 та кроком 2 (який вказує, що об'єднані регіони не перекриваються). Перший агрегувальний шар з’єднується із згортковим шаром, який щойно був створений. Використаємо метод max_pooling2d ()для побудови шару, який виконує максимальне об'єднання з фільтром необхідного розміру:

```
with tf.variable_scope('pool1') as scope:
    pool1 = tf.layers.max_pooling2d(
        inputs=conv1,
        pool_size=[2, 2],
        strides=2
    )
```

Конволюційний шар №2: застосовується 64 фільтрів 5x5 з функцією активації ReLU

```
# [Convolution #2]
with tf.variable_scope('conv2') as scope:
    filters_count = 64
    kernel_size = 5

    conv2 = tf.layers.conv2d(
        inputs=pool1,
        filters=filters_count,
        kernel_size=kernel_size,
        activation=tf.nn.relu,
        kernel_initializer=tf.random_normal_initializer(),
        padding='SAME')
```

Агрегувальний шар №2: Знову ж таки, виконує максимум об'єднання з фільтром 2x2 та кроком 2

```
with tf.variable_scope('pool2') as scope:
    pool2 = tf.layers.max_pooling2d(
        inputs=conv2,
        pool_size=[2, 2],
        strides=2
    )
```

Повнозв’язний шар №1 з коефіцієнтом регуляризації відсіву 0,4 (ймовірність 0,4, що будь-який даний елемент буде викинути під час навчання).

```
# Dense Layer
with tf.variable_scope('dense') as scope:
    pool2_flat = tf.reshape(pool2, [-1, 64*62*62])
    dense = tf.layers.dense(inputs=pool2_flat, units=64*62*62, activation=tf.nn.relu)
    dropout = tf.layers.dropout(
        inputs=dense, rate=DROP_OUT_RATE, training=mode == tf.estimator.ModeKeys.TRAIN)
    # Logits Layer
    logits = tf.layers.dense(inputs=dropout, units=10)
```

Повнозв'язний шар № 2 (Ligits Layer): 3 нейрони, один для кожного класу цільового класу (0-2).

Для навчання та оцінки необхідно визначити функцію втрат, яка визначає, наскільки точно прогнози моделі співпадають з цільовими класами. Функція помилки:

```
# Calculate Loss (for both TRAIN and EVAL modes)
loss = tf.losses.sparse_softmax_cross_entropy(labels=labels, logits=logits)
```

Модуль tf.layers містить методи створення кожного з трьох вищезазначених типів шарів:

- conv2d(). Конструює двовимірний згортковий шар. В якості аргументів приймається кількість фільтрів, розмір ядра фільтра, підкладка та функція активації.
- max_pooling2d(). Створює двомірний агрегувальний шар, використовуючи алгоритм максимального об'єднання. Приймає розмір фільтра та крок як аргументи.

- dense(). Конструює щільний шар. Враховує кількість нейронів і функцію активації як аргументи.

Кожен з цих методів приймає тензор як вхід і повертає трансформований тензор у вигляді виходу. Це полегшує підключення одного шару до іншого: просто виведіть вихід з одного методу створення шару та подайте на інший шар.

Обчислення, які використовуються для TensorFlow, наприклад, тренування багатошарової нейронної мережі, можуть бути складними та заплутаними. Для полегшення розуміння, налагодження та оптимізації програм TensorFlow включено набір інструментів візуалізації, що називаються TensorBoard. TensorBoard можен бути використаним для візуалізації графіків TensorFlow, побудови кількісних показникиків щодо виконання графіка та відображення додаткових даних, таких як зображення, які проходять через нього. Використаємо інструмент Tensorboard для візуалізації отриманого обчислювального графу (рис 9.1.1 та рис.9.1.2):

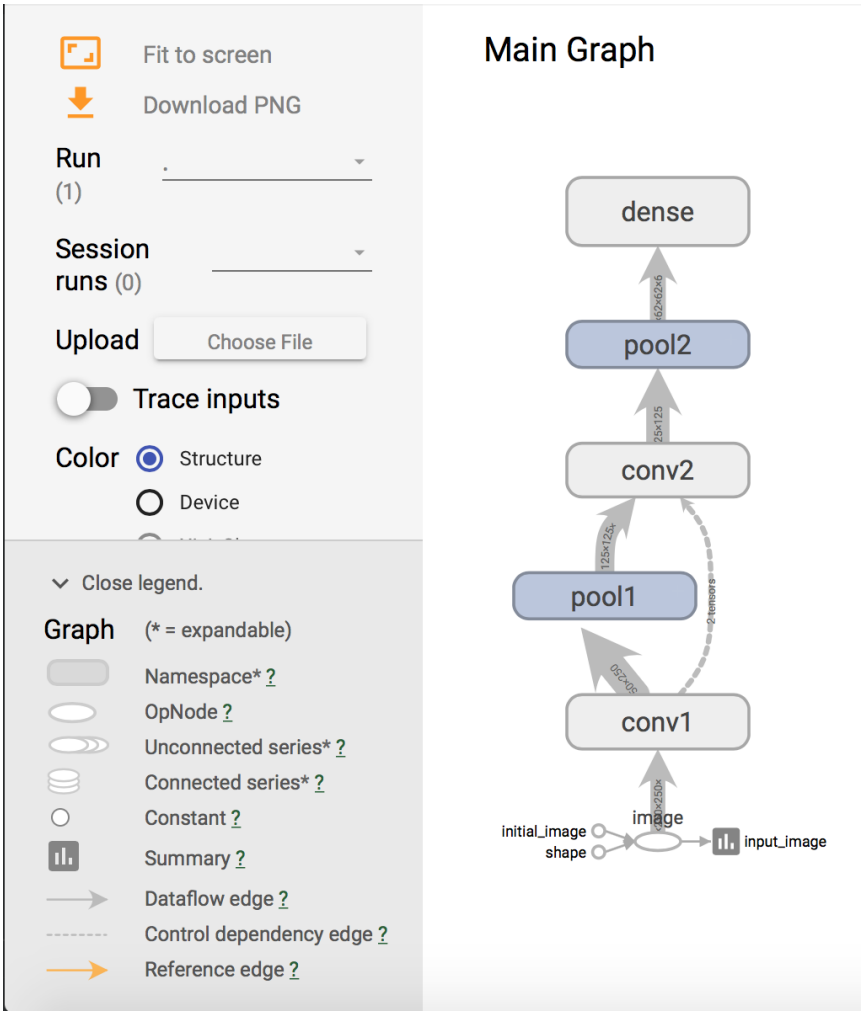

Рис.9.1.1 Обчислювальний граф згорткової мережі

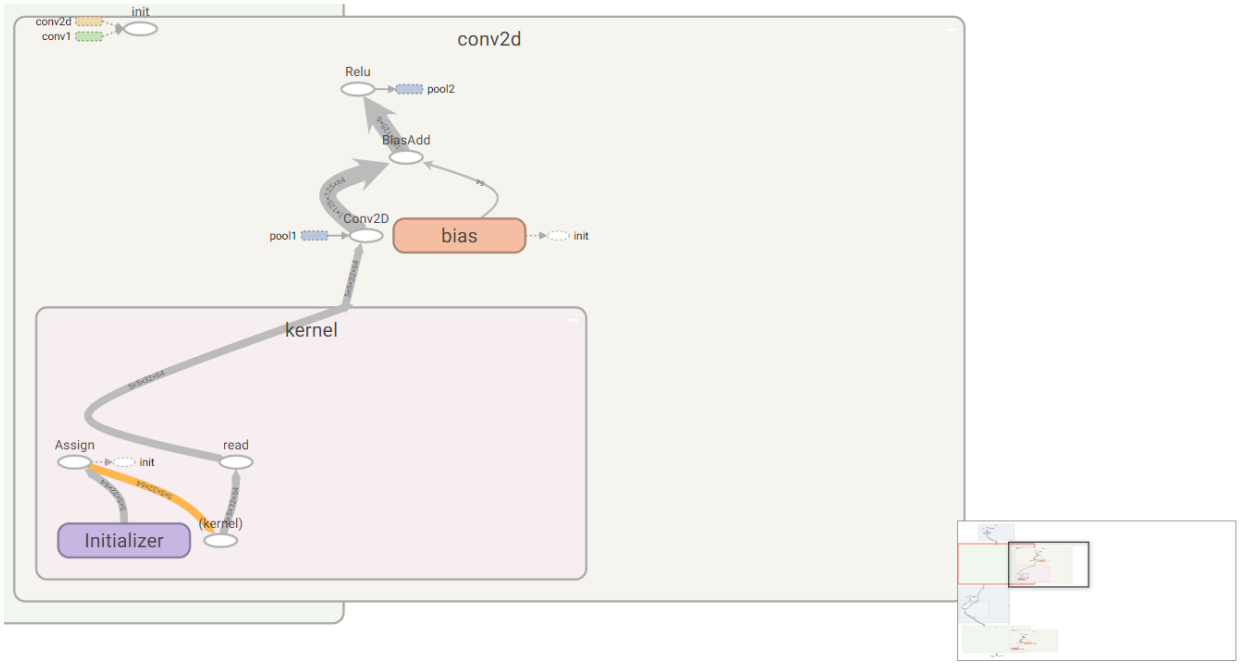

Рис.9.1.2 Обчислювальний граф для шару згортки

9.2 Технологія Docker

Прототип діагностичної системи класифікації шкірних захворювань розгортається в хмарі в якості сервісу. Це дозволяє замовнику програмного забезпечення не підтримувати окремо інфраструктуру, а придбати ліцензію на використання сервісу, а відповідальність за підтримку інфраструктури бере на себе хмарний провайдер.

Docker є найпопулярнішою платформою, що використовує технологію контейнеризації. Платформа Docker вирішує три основні проблеми розгортання сервісів:

- доставка коду на сервер;
- запуск коду;
- однаковість оточення.

Docker дозволяє ізолювати сервіси від інфраструктури, таким чином досягається можливість доставляти їх набагато швидше. Docker дозволяє управляти інфраструктурою за допомогою тих же принципів, які використовуються при управлінні додатками. При використанні інструментів Docker для доставки, тестування і розгортання, значно скорочується затримка між фіксацією нового коду в системі контролю версій і запуском його на сервері промислової експлуатації [21].

Docker надає можливість упаковувати і запускати сервіси в ізольованому оточенні, яке і називається контейнером. Дана ізольованість і безпека дозволяє запускати кілька контейнерів на одному хості одночасно. Контейнери легковісні оскільки не вимагають роботи гіпервізора. Вони запускаються безпосередньо на ядрі машини-хоста. Таким чином досягається можливість запуску більшої кількості контейнерів, ніж віртуальних машин, на одному і тому ж фізичному обладнанні. Також, Docker контейнери можна запускати в самих віртуальних машинах.

Docker надає наступні інструменти і платформу для управління життєвим циклом контейнерів. За допомогою Docker відбувається упаковка додатків і їх компонентів в контейнер. В екосистемі Docker, контейнер стає

атомарним юнітом для поширення і тестування програми. Після розробки, додаток може бути розгорнуто на сервері промислової експлуатації вручну або за допомогою оркестровщика контейнерів. Дана техніка розгортання однакова незалежно від того де розгортається додаток в промислову експлуатацію, будь то в локальному дата центрі, хмарному провайдера або в гібридному оточенні.

Docker оптимізує життєвий цикл розробки, дозволяючи розробникам працювати в стандартизованому оточенні. При його використанні, оточення при розробці нічим не відрізняється від оточення в промисловій експлуатації.

Контейнери ідеально підходять для безперервної інтеграції або безперервного постачання. Звичайний сценарій розробки програмного забезпечення зводиться до отримання якогось артефакту, який будується після будь-якої зміни вихідного коду програми. При використанні докер, цим артефактом може бути контейнер. Розробники або інженери з тестування використовують докер контейнери для розгортання додатків на тестове оточення і запускають автоматичні або ручні тести. Якщо під час перевірок виявляється баг, даний контейнер може бути розгорнуто в оточенні розробки, виправлений і перенесений назад в тестове оточення, для повторної перевірки. Якщо всі перевірки успішно завершені досить доставити контейнер з виправленнями в промислове оточення.

Контейнерна платформа Docker забезпечує високу переносимість. Контейнер може виконуватися на робочій станції розробника, фізичних або віртуальних машинах в дата центрі, в інфраструктурі хмарного провайдера. Переносимість докера, а також його легкість дозволяють динамічно міняти завантаження, збільшуючи або зменшуючи кількість додатків і сервісів, практично в реальному часі.

Docker легкий і швидкий. Він надає життєздатну, економічно вигідну альтернативу віртуальним машинам на основі гіпервізора. За допомогою докер можна використовувати більше обчислювальних потужностей, оскільки ці ресурси не витрачаються на обслуговування гіпервізора.

Docker використовує клієнт-серверну архітектуру (рис.9.2.1)

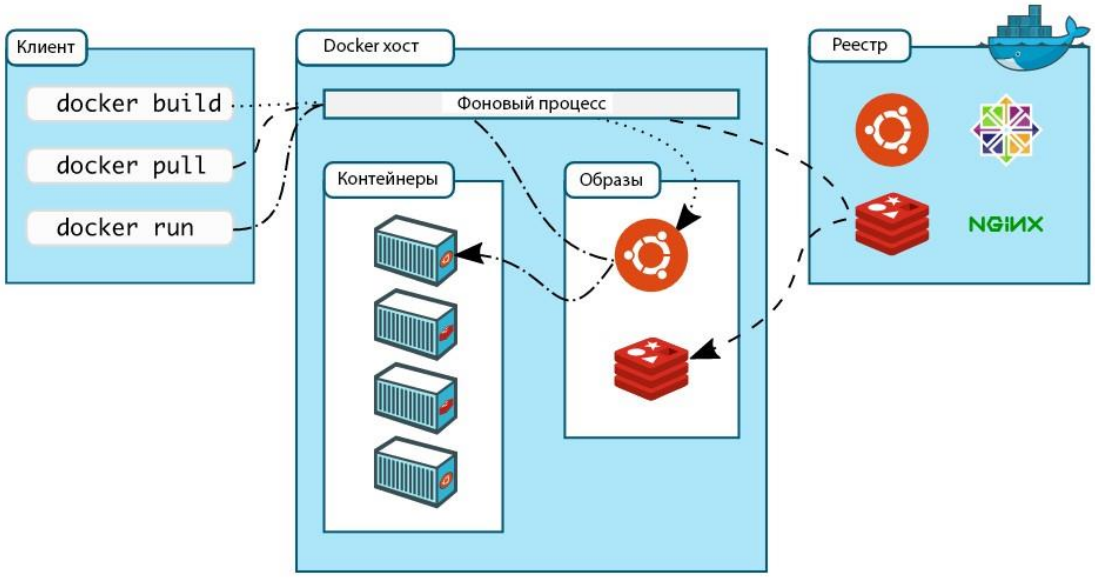

Рис.9.2.1 Архітектура Docker

Docker клієнт взаємодіє з фоновим процесом - сервером, який в свою чергу запускає контейнери. Клієнт і фоновий процес можуть виконуватися в одній операційній системі, також є можливість підключити клієнта до віддаленого фонового процесу. Докер клієнт і фоновий процес взаємодіють через REST API поверх сокетів або через мережевий інтерфейс. Фоновий процес Docker (dockerd) приймає запити і управляє об'єктами Docker. Фоновий процес може також взаємодіяти з іншими фоновими процесам. Фоновий процес управляє наступними об'єктами:

- образи;
- контейнери;
- мережеве взаємодія;
- томи.

Docker клієнт основний спосіб взаємодії з сервером. При виконанні команди клієнт відправляє цю команду фонового процесу, який в свою чергу її виконує. Docker клієнт може взаємодіяти з безліччю різних серверів.

Реєстр контейнерів зберігає образи. Даний реєстр може бути, як публічним, так і приватним. Docker Hub є офіційним відкритим реєстром, в якому зберігаються офіційні образи від різних провайдерів технологій,

наприклад, ubuntu або nginx. Додавати образи в публічний реєстр може будь-хто. За замовчуванням Docker налаштований на пошук образів в цьому публічному реєстрі. Також є можливість створювати приватні реєстри образів.

При використанні команд docker pull або docker run, необхідні образи будуть завантажені з сконфігурованого реєстру. При використанні команди docker push, вказаний образ буде поміщений в реєстр.

9.3 Підготовка сервісу до розгортання в хмарі

Для розгортання прототипу діагностичної системи класифікації шкірних захворювань необхідно зібрати Docker образ, який містить всі пакети та залежності для мови програмування Python та програмної платформи для багаторівневого навчання Tensorflow, а також код самого прототипу. Використаємо пакетний менеджер для операційної системи Homebrew, який забезпечує ізольоване середовище для встановлення пакетів та управління їх залежностями. Для початку встановимо програмне забезпечення Docker за допомогою команди

brew install docker docker-compose

Перевіримо результат встановлення програмного забезпечення за допомогою команди

docker –version

Отримаємо в результаті встановлену версію програмного забезпечення:

Vadyms-MacBook-Pro:docs vadym$ docker --version

Docker version 17.12.0-ce, build c97c6d6

В публічних репозиторіях Docker Hub наразі підтримується два образи Tensorflow:

- tensorflow / tensorflow - TensorFlow з усіма залежностями - тільки для CPU;
- tensorflow / tensorflow: latest-gpu - TensorFlow з усіма залежностями та підтримкою для NVidia CUDA, тобто з можливістю запуску навчання нейронної мережі на відеокартах.

Запустимо контейнер без підтримки GPU (так як немає можливості використання потужних дискретних відеокарт) :

docker run -it -p 8888:8888 tensorflow/tensorflow

Для запуску контейнера з підтримкою GPU необхідно попереднє встановлення NVidia драйверів та nvidia-docker інструменту наступним чином:

nvidia-docker run -it -p 8888:8888 tensorflow/tensorflow:latest-gpu

За посиланням http://localhost:8888 сервіс Jupyter Notebook, який дозволяє створювати Python модулі в рамках даного образу, імпортувати необхідні залежності (рис. 9.3.1).

Рис. 9.3.1 Jupyter Notebook

IPython представляє собою потужний інструмент для роботи з мовою Python. Базові компоненти IPython – це інтерактивна оболочка для широкого набору можливостей і ядра для Jupyter. Ноутбук Jupyter є графічною веб-оболонкою для IPython, яка розширює ідею консольного підходу до інтерактивних розрахунків.

Основні відмінні риси даної платформи – це комплексна інтроспекція об'єктів, збереження історії введення протягом усіх сеансів, кешування вихідних результатів, що розширюється система «магічних» команд, логування сесії, додатковий командний синтаксис, підсвічування коду, доступ до системної оболонці, стикування з PDB відладчиком і Python профайлер.

IPython дозволяє підключитися безлічі клієнтів до одного комп'ютера і, завдяки своїй архітектурі, може працювати в паралельному кластері. У

Jupyter ноутбук можна розробляти, документувати та запускати програми на мові Python, він складається з двох компонентів: веб-додаток, що запускається в браузері, і ноутбуки – файли, в яких можна працювати з вихідним кодом програми, запускати його, вводити і виводити дані і т.п. Веб додаток дозволяє:

- Редагувати Python код в браузері з підсвічуванням синтаксису, автовідступами та автодоповненнями;
- запустити код в браузері;
- відображати результати обчислень із медіа представленням (схеми, графіки);
- працювати з мовою розмітки Markdown і LaTeX.

Ноутбуки - це файли, в яких зберігаються вихідний код, вхідні та вихідні дані, отримані в межах сесії. Фактично, вони є записом вашої роботи, але при цьому дозволяє заново виконати код, присутній на ньому. Ноутбуки можна експортувати в форматах PDF, HTML.

Оскільки Docker контейнер є тимчасовим екземпляром, який зазвичай ізольований від всього обчислювального середовища. Це дає можливість легко переносити контейнер з одного середовища в інше, але після написання вихідного коду прототипу (Додаток А та Додаток Б) необхідно зберегти отримані результати та модулі в рамках даного образу з можливістю його подальшого запуску. Для цього використаємо утиліту командного рядка *docker commit*, яка дозволяє зберегти стан контейнеру як окремий докер образ, який можна потім запустити окремо. Отриманий образ, як і будь-який інший контейнер, можна легко потім перенести в інше обчислювальне середовище. Необхідно на машині, на якій запущено Docker, дізнатися ID запущеного контейнеру за допомогою команди *docker ps:*

Vadyms-MacBook-Pro:masterdiploma vadym$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

fbe73b1bb2b1 gcr.io/tensorflow/tensorflow "/run_jupyter.sh --a…" 31 minutes ago Up 31 minutes 0.0.0.0:6006->6006/tcp, 0.0.0.0:8888->8888/tcp clever_mestorf

Збережемо даний образ за допомогою команди *docker commit fbe73b1bb2b1 vadym/gcr.io/tensorflow/tensorflow*. Також можна опублікувати отриманий артефакт у вигляді збереженого Docker образу в приватному регістрі Docker Hub за допомогою команди:

docker push fbe73b1bb2b1 vadym/gcr.io/tensorflow/tensorflow

Отриманий Docker образ повністю готовий до розгортання в рамках будь-якого хмарного провайдера.

9.4 Діагностика отриманої моделі

Розглянемо поняття відносної ентропії (або відстані Кульбака – Лейблера), що представляє собою міру різниці між двома ймовірнісними розподілами P і Q [16]. Як правило, вважається, що P – це істинний розподіл, а Q – його наближення, тоді відстань Кульбака-Лейблера служить оцінкою якості наближення. У випадку, коли P і Q – дискретні випадкові величини на дискретній множині $X = \{x_1, \dots, x_N\}$, відстань Кульбака – Лейблера виглядає наступним чином [8]:

$$KL(P||Q) = \sum_i p(x_i)\, log\, \frac{p(x_i)}{q(x_i)},$$

де $p(x_i)$ і $q(x_i)$ – власне ймовірності результату x_i. В якості функції помилки використовується перехресна ентропія:

$$H(p,q) = E_p[-log\ q] = H(p) + D_{KL}(p||q)$$

Для дискретних p і q маємо:

$$H(p,q) = -\sum_{\forall x} p(x) \log q(x)$$

Отже, проведемо навчання отриманої моделі і відобразимо для кожної епохи точність на навчальній вибірці, на валідаційній вибірці та значення функції помилки (рис. 9.4.1).

```
Epoch 1 --- Training Accuracy:  56.2%, Validation Accuracy:  75.0%, Validation Loss: 0.615
Epoch 2 --- Training Accuracy:  56.2%, Validation Accuracy:  56.2%, Validation Loss: 0.679
Epoch 3 --- Training Accuracy:  56.2%, Validation Accuracy:  43.8%, Validation Loss: 0.712
Epoch 4 --- Training Accuracy:  68.8%, Validation Accuracy:  31.2%, Validation Loss: 0.713
Epoch 5 --- Training Accuracy:  68.8%, Validation Accuracy:  68.8%, Validation Loss: 0.645
Epoch 6 --- Training Accuracy:  75.0%, Validation Accuracy:  56.2%, Validation Loss: 0.697
Epoch 7 --- Training Accuracy:  81.2%, Validation Accuracy:  68.8%, Validation Loss: 0.623
Epoch 8 --- Training Accuracy:  75.0%, Validation Accuracy:  50.0%, Validation Loss: 0.670
Epoch 9 --- Training Accuracy:  75.0%, Validation Accuracy:  43.8%, Validation Loss: 0.753
Epoch 10 --- Training Accuracy:  75.0%, Validation Accuracy:  43.8%, Validation Loss: 0.781
Epoch 11 --- Training Accuracy:  75.0%, Validation Accuracy:  62.5%, Validation Loss: 0.605
Epoch 12 --- Training Accuracy:  75.0%, Validation Accuracy:  50.0%, Validation Loss: 0.735
Epoch 13 --- Training Accuracy:  75.0%, Validation Accuracy:  81.2%, Validation Loss: 0.441
Epoch 14 --- Training Accuracy:  75.0%, Validation Accuracy:  62.5%, Validation Loss: 0.604
Epoch 15 --- Training Accuracy:  75.0%, Validation Accuracy:  43.8%, Validation Loss: 0.813
Epoch 16 --- Training Accuracy:  75.0%, Validation Accuracy:  43.8%, Validation Loss: 0.910
Epoch 17 --- Training Accuracy:  87.5%, Validation Accuracy:  62.5%, Validation Loss: 0.629
Epoch 18 --- Training Accuracy:  87.5%, Validation Accuracy:  56.2%, Validation Loss: 0.799
Epoch 19 --- Training Accuracy:  87.5%, Validation Accuracy:  93.8%, Validation Loss: 0.296
Epoch 20 --- Training Accuracy:  87.5%, Validation Accuracy:  81.2%, Validation Loss: 0.446
Epoch 21 --- Training Accuracy:  87.5%, Validation Accuracy:  50.0%, Validation Loss: 1.000
Epoch 22 --- Training Accuracy:  87.5%, Validation Accuracy:  56.2%, Validation Loss: 0.914
Epoch 23 --- Training Accuracy:  87.5%, Validation Accuracy:  68.8%, Validation Loss: 0.771
Epoch 24 --- Training Accuracy:  87.5%, Validation Accuracy:  56.2%, Validation Loss: 0.964
Epoch 25 --- Training Accuracy:  93.8%, Validation Accuracy:  93.8%, Validation Loss: 0.259
Epoch 26 --- Training Accuracy:  93.8%, Validation Accuracy:  81.2%, Validation Loss: 0.360
Epoch 27 --- Training Accuracy:  93.8%, Validation Accuracy:  50.0%, Validation Loss: 1.379
Epoch 28 --- Training Accuracy:  93.8%, Validation Accuracy:  50.0%, Validation Loss: 1.011
Epoch 29 --- Training Accuracy:  93.8%, Validation Accuracy:  62.5%, Validation Loss: 0.997
Epoch 30 --- Training Accuracy:  93.8%, Validation Accuracy:  50.0%, Validation Loss: 1.423
Epoch 31 --- Training Accuracy:  93.8%, Validation Accuracy:  93.8%, Validation Loss: 0.225
Epoch 32 --- Training Accuracy:  93.8%, Validation Accuracy:  81.2%, Validation Loss: 0.316
```

Рис.9.4.1 Результати навчання класифікаційної моделі

9.5 Висновки

В рамках даного розділу було виконано практичну частину дипломного проекту, а саме реалізацію графу обчислень з використанням мови програмування Python та програмного пакету Tensorflow. Було здійснено підготовку вхідних даних для навчання багарівневої мережі, а саме реалізовано Spark джобу для розподіленої обробки великого набору зображень з метою їх приведення до єдиного розміру.

Прототип сервісу автоматизованої класифікаї захворювань шкіри було підготовлено до розгортання в рамках інфраструктури хмарних провайдерів, а саме було створено Docker образ з необхідними пакетами, програмними модулями та залежностями.

Для діагностування отриманої класифікаційної моделі використано таку функцію помилки як перехресна ентропія, що виражається через відстань Кульбака-Лейблера, тобто міру різниці між двома ймовірнісними розроділами. Отримана модель в результаті показала високу точність

класифікації на валідаційній вибірці, а саме 81.2% , що підтверджує ефективність використання алгоритмів багаторівневого навчання для допомоги дерматологам в діагностування шкірних захворювань.

10 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ “СЕРВІС ДЛЯ КЛАСИФІКАЦІЇ ШКІРНИХ ЗАХВОРЮВАНЬ”

10.1 Опис ідеї проекту

У даному розділі описано економічне обґрунтування розроблення стартап-проекту на тему “Сервіс для класифікації шкірних захворювань”. Захворювання шкіри належать до розповсюджених медичних проблем. Кількість таких захворювань постійно зростає, незважаючи на розвиток медичної галузі. Рак шкіри є поширеним злоякісним новоутворенням і займає друге рангове місце у структурі онкологічної захворюваності населення України. Такі хвороби діагностуються візуально, починаючи з клінічних обстежень, що можуть супроводжуватись дерматоскопічним аналізом, біопсією та гістопатологічною експертизою. Особливий інтерес представляє автоматизована класифікація захворювань шкіри (як доброякісних, так і злоякісних) на базі зображення ураженої ділянки тіла.

Таблиця 10.1 - Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
Ідея полягає у створенні сервісу автоматизованої класифікації захворювань шкіри (як доброякісних, так і злоякісних) на базі зображення ураженої ділянки тіла.	1. Допоміжний сервіс для медиків-дерматологів.	Допомога в прийнятті правильного рішення при діагностуванні хвороби. Система аналізує масив знімків раніше діагностованих хвороб для здійснення класифікації.
	2. Сервіс самостійної попередньої діагностики для пацієнтів.	Можливість проведення попередньої діагностики в домашніх умовах (маючи лише пристрій з фотокамерою). Можливість завантажувати в систему фотографії ураженої ділянки тіла та результати аналізів для лікаря.

Описаний сервіс може бути реалізованим у вигляді веб-додатку для медиків-дерматологів та у вигляді застосування для мобільних платформ для кінцевих пацієнтів. Ним можна буде користуватись в будь-який зручний спосіб при

наявності підключення до мережі Internet. Інтеграція сервісу з електронним кабінетом пацієнта, на відміну від самостійних сервісів-замінників, дозволить налагодити швидку взаємодію пацієнта з власним лікарем-дерматологом.

Визначений перелік слабких, сильних та нейтральних характеристик та властивостей ідеї потенційного товару є підґрунтям для формування його конкурентоспроможності. Аналіз потенційних техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів передбачає:

- визначення переліку техніко-економічних властивостей та характеристик ідеї
- визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводиться збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;
- проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні).

Таблиця 10.2 - Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	*Техніко-економічні характеристики ідеї*	*(Потенційні) товари/концепції конкурентів*				*W (слабка сторона)*	*N (нейтральна сторона)*	*S (сильна сторона)*
		Мій проект	*Конкурент1*	*Конкурент2*	*Конкурент3*			
1.	Форма виконання	Сервіс і мобільний клієнт	Додаток	Веб-сервіс	Веб-сервіс			+
2.	Собівартість	Низька	Висока	Середня	Середня			+

3.	Наявність адміністратора для налаштуван-ня	Не треба	Не треба	Пот-рібно	Пот-рібно			+
4.	Наявність інтернету	Треба	Треба	Треба	Не треба		+	
5.	Крос-платформеність	Ні	Так	Ні	Ні	+		

Сильною стороною даного проекту є форма виконання у вигляді сервісу, це дає можливість не витрачати бюджет на встановлення апаратного комплексу та програмного забезпечення і дозволяє обійтись без адміністратора для налаштувань. Необхідно лише придбати ліцензію на певний строк використання даного продукту. Слабкою стороною проекту є кросплатформеність, так як необхідним є реалізація мобільного клієнта під різні платформи для кінцевих пацієнтів. Всі інші характеристики є нейтральними, тому даний проект можна вважати конкурентоспроможним.

10.2 Технологічний аудит ідеї проекту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару). Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових:

- за якою технологією буде виготовлено товар згідно ідеї проекту?
- чи існують такі технології, чи їх потрібно розробити/доробити?
- чи доступні такі технології авторам проекту?

Таблиця 10.3 – Технологічна здійсненність ідеї проекту

№ п/п	*Ідея проекту*	*Технології її реалізації*	*Наявність технології*	*Доступність технології*
1.	Створення сервісу	Python (Tensorflow) CUDA	Наявна	Безкоштовна, доступна
		Lua, Torch	Наявна	Платна, доступна

Обрана технологія реалізації ідеї проекту: для створення сервісу обрана технологія Python (Tensorflow), яка є безкоштовною та доступною.

Для імплементації проекту були обрані технології Python (Tensorflow) та CUDA. Основним критерієм вибору стеку технологій є розмір активної спільноти розробників, проте не менш важливим є той факт, що такий стек є безкоштовним. Відсутність необхідності вивчати нову мову програмування Lua дає можливість залучення великої аудиторії Python розробників.

10.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку проводимо аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку.

Таблиця 10.4 - Попередня характеристика потенційного ринку стартап-проекту

№ п/п	*Показники стану ринку (найменування)*	*Характеристика*
1.	Кількість головних гравців, од	3
2.	Загальний обсяг продаж, грн/ум.од	20000 грн./ум.од
3.	Динаміка ринку (якісна оцінка)	Зростає
4.	Наявність обмежень для входу (вказати характер обмежень)	Немає
5.	Специфічні вимоги до стандартизації та сертифікації	Немає
6.	Середня норма рентабельності в галузі (або по ринку), %	R = (3000000 * 100) / (1000000 * 12) = 25%

Середня норма рентабельності в галузі (або по ринку) є вищою за банківські відсотки на вкладення. Враховуючи також відсутність обмежень для входу в галузь та зростаючу динаміку ринку, можна зробити висновок що ринок є

привабливим для входження за попереднім оцінюванням і є сенс вкласти кошти саме в даний проект.

Таблиця 10.5 - Характеристика потенційних клієнтів стартап-проекту

№ п/п	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
1.	Сервіс автоматизованої класифікації захворювань шкіри (як доброякісних, так і злоякісних) на базі зображення ураженої ділянки тіла.	Медики-дерматологи та пацієнти, що мають захворювання шкіри.	Медики-дерматологи потребують веб-сервіс як допоміжний засіб для правильного діагностування хвороби. В той час як кінцевим пацієнтам необхідною є можливість мати мобільний застосунок для попередньої самостійної діагностики та інтеграцію з електронним кабінетом з можливістю поділитися з лікарем фотографіями та результатами аналізів.	Рішення повинне бути зручним у користуванні, надійним, швидким

Згідно проведеної характеристики потенційних клієнтів стартап-проекту випливає, що аудиторія даного програмного забезпечення розбивається на дві групи користувачів, з одного боку це медики-дерматологи, яким необхідна допомога в прийнятті рішень при діагностуванні захворювань, а з іншого боку – пацієнти, яким важливо мати змогу провести в домашніх умовах попередню діагностику. Обидві групи об’єднані спільним попитом у сервісі з високою точністю класифікації та зручності користування.

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (табл. № 6-7). Фактори в таблиці подавати в порядку зменшення значущості. Ринкові можливості - це сприятливі обставини, які підприємство може використовувати для отримання переваг. Слід зазначити, що можливостями з погляду SWOT-аналізу є не всі можливості, які існують на ринку, а тільки ті, які можна використовувати. Ринкові загрози - події, настання яких може несприятливо вплинути на підприємство.

Таблиця 10.6 – Фактори загроз

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1.	Зміна потреб користувачів	Користувачам необхідний сервіс з іншим функціоналом	Передбачити архітектурно можливість додавання нового функціоналу до створюваного додатку
2.	Економічний	Подорожчання послуг провайдера інфраструктури.	Оптимізація програмного продукту для зменшення кількості мережевого трафіку, пам’яті та процесорного часу, що споживаються.
3.	Конкуренція	Вихід на ринок великої компанії	1) Вихід з ринку 2) Запропонувати великій компанії поглинути себе 3) Передбачити додаткові переваги власного ПЗ для того, щоб повідомити про них саме після виходу міжнародної компанії на ринок
4.	Науковий	Поява нової архітектури нейронної мережі, що продемонструє вищу точність класифікації	Передбачити можливість заміни архітектури класифікатора
5.	Зміна податкової політики держави	Зростання розміру податків	Підвищення вартості ліцензії на використання продукту

Ринкові загрози – події, настання яких може несприятливо вплинути на підприємство. Серед найвпливовіших загроз даного проекту - вихід на ринок великої компанії, що здатна скласти конкуренцію, а також зміна податкової

політики держави. Поява конкуренції може стати рушієм розвитку функціональної частини проекту, тобто перетворитися із загрози в можливість. В той чай як зростання розміру податків неодмінно приведе до підвищення вартості ліцензії на використання продукту.

Таблиця 10.7 – Фактори можливостей

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1.	Зростання можливостей потенційних покупців	Ріст зацікавленості до продукту серед інших груп користувачів з різним рівнем технічної грамотності	Запропонувати їм свої послуги
2.	Зниження довіри до конкурентів	У додатку конкурента нещодавно стався збій і протягом тижня система неправильно функціонувала	При виході на ринок звертати увагу покупців на надійність нашого сервісу
3.	Попит	Більш специфічні вимоги до програмного продукту	1) Підтримка продукту 24/7 2) Пропозиція нових продуктових рішень.
4.	Реформа навчальної програми для студентів-медиків	Впровадження нової навчальної програми для студентів-дерматологів, що передбачає ознайомлення з технологією автоматизованої класифікації захворювань шкіри.	Надання можливості отримати безкоштовну пробну ліцензію з метою проведення навчальних робіт в даній програмній системі.
5.	Технологічний	Поява на ринку більш точної архітектури класифікатора	Впровадження нової архітектури в програмне середовище та її подальше тестування.

У Таблиці 10.7 наведено фактори можливостей ринкового впровадження проекту, серед яких найбільш вагомими є: технологічний (завдяки можливості впровадження нової архітектури в програмне середовище, що дозволить підвищити точність класифікаційної моделі) та реформа навчальної програми для студентів-медиків (даний фактор сприяє впровадженню даної системи в якості навчальної з метою допомоги студентам-дерматологам у здобутті практичних навичок діагностування захворювань шкіри).

Надалі проводиться аналіз пропозиції: визначаються загальні риси конкуренції на ринку (табл. 8).

Таблиця 10.8 - Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)*
1. Вказати тип конкуренції: - досконала	Існує 3 фірми-конкуренти на ринку	Врахувати ціни конкурентних компаній на початкових етапах створення бізнесу, реклама (вказати на конкретні переваги перед конкурентами)
2. За рівнем конкурентної боротьби: - міжнародний	Всі компанії – з інших країн	Додати можливість вибору мови ПЗ, щоб легше було у майбутньому вийти на міжнародний ринок
3. За галузевою ознакою: - внутрішньогалузева	Конкуренти мають ПЗ, який використовується лише всередині даної галузі	Створити основу ПЗ таким чином, щоб можна було легко його переробити для використання у інших галузях
4. Конкуренція за видами товарів: - товарно-видова	Види товарів є однаковими, а саме - програмне забезпечення	Створити ПЗ, враховуючи недоліки конкурентів
5. За характером конкурентних переваг: - нецінова	Вдосконалення технології створення ПЗ, щоб собівартість була нижчою	Використання менш дорогих технологій для розробки, ніж використовують конкуренти
6. За інтенсивністю: - не марочна	Бренди відсутні	-

У Таблиці 10.8 наведено ступеневий аналіз конкуренції на ринку, де було визначено особливості конкурентного середовища та їх вплив а діяльність підприємства. Однією з найбільш важливих дій компанії для досягнення конкурентоспроможності є необхідність створити основу ПЗ таким чином, щоб можна було легко переробити дане ПЗ для використання у інших галузях.

Після аналізу конкуренції проводиться більш детальний аналіз умов конкуренції в галузі. *М. Портер* вирізняє п‘ять основних факторів, що впливають на привабливість вибору ринку з огляду на характер конкуренції.

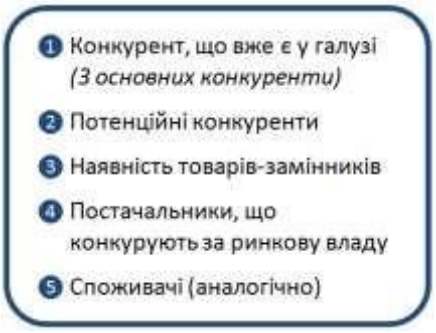

Рис. 10.1 – Складові моделі 5 сил М. Портера

Сильні позиції компанії за кожним з факторів означають її можливості забезпечити необхідні темпи обороту капіталу та її здатність впливати на інших агентів ринку, диктуючі їм власні умови співпраці.

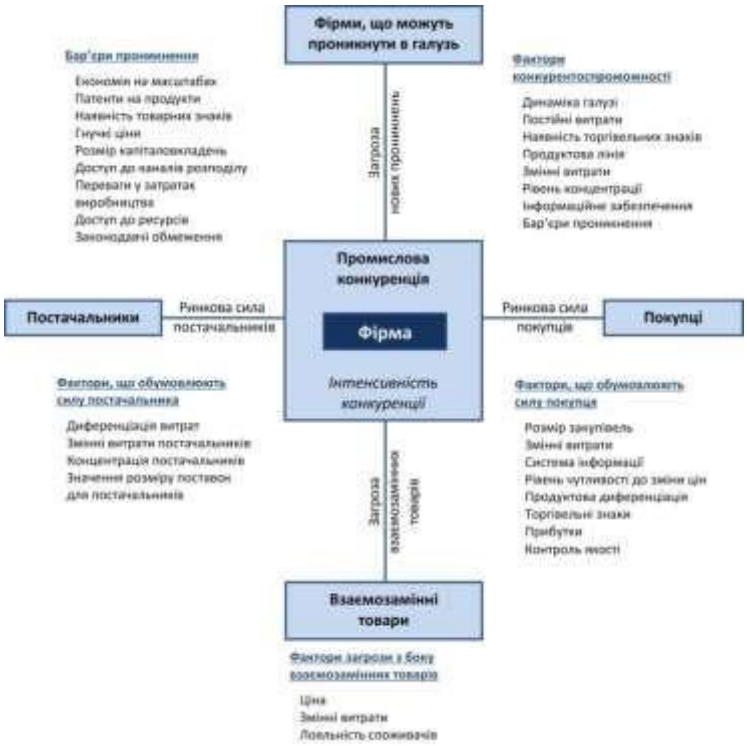

Рис. 10.2 – Модель 5 сил М. Портера для аналізу конкуренції в галузі

Характеристики факторів моделі відрізняються для різних галузей та змінюються із часом. Сила кожного фактору є функцією від структури галузі та її техніко-економічних характеристик. На основі аналізу складових моделі 5 сил М. Портера розробляється перелік факторів конкурентоспроможності для певного ринку.

Таблиця 10.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	*Прямі конкуренти в*	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*

	галузі				
	Навести перелік прямих конкурентів	*Визначити бар'єри входження в ринок*	*Визначити фактори сили постачальників*	*Визначити фактори сили споживачів*	*Фактори загроз з боку замінників*
Висновки	Існує 3 конкуренти на ринку. Найбільш схожим за виконанням є конкурент 2, так як його рішення також представлене у вигляді Веб сервісу.	Можливості для входу на ринок є, бо наше рішення спрощує та пришвидшує роботу медика-дерматолога.	Постачальники відсутні.	Важливим для користувача є зручність у користуванні	Товари-замінники можуть використати більш дешеву технологію створення ПЗ та зменшити собівартість товару

Було здійснено аналіз конкуренції в галузі за М. Портером, в результаті чого було визначено, що існує 3 конкуренти на ринку. Найбільш схожим за виконанням є конкурент 2, так як його рішення також представлене у вигляді веб-додатку, але можливості для входу на ринок є, бо рішення спрощує та пришвидшує роботу медика-дерматолога.

За результатами аналізу таблиці робиться висновок щодо принципової можливості роботи на ринку з огляду на конкурентну ситуацію. Також робиться висновок щодо характеристик (сильних сторін), які повинен мати проект, щоб бути конкурентоспроможним на ринку. Другий висновок враховується при формулюванні переліку факторів конкурентоспроможності у п. 3.6. 3.6. На основі аналізу конкуренції, проведеного в п. 3.5 (табл. 9), а також із урахуванням характеристик ідеї проекту (табл. 2), вимог споживачів до товару (табл. 5) та факторів маркетингового середовища (табл. № 6-7) визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлюється за табл. 10.

Таблиця 10.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	*Фактор конкурентоспроможності*	*Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)*
1.	Використання ПЗ у вигляді сервісу	Дозволяє уникнути фінансових витрат і складнощів підтримки власної інфраструктури для програмної платформи.
2.	Простота інтерфейсу користувача	Користувач має лише завантажити фото враженої ділянки шкіри і запустити програму на виконання.
3.	Висока точність класифікації	Програмне середовище дозволяє отримати високу точність класифікаційної моделі. Також архітектура ПЗ спроектована таким чином, що можливим є зміна структури та інші оптимізації моделі класифікації.
4.	Хмарне розгортання	Дозволяє звертатись до системи автоматизованої класифікації шкірних захворювань як до сервісу
5.	Вертикальне масштабування	Можливість гнучкого масштабування за допомогою оптимізації апаратних компонентів, заміни CPU на відеокарти.

У Таблиці 7 наведено обґрунтування факторів конкурентоспроможності, серед яких: форма імплементації у вигляді сервісу, висока точність класифікації, можливість хмарного розгортання та можливість вертикального масштабування системи. Було також наведено обґрунтування цих факторів.

За визначеними факторами конкурентоспроможності (табл. 4.10) проводиться аналіз сильних та слабких сторін стартап-проекту (табл. 4.11). Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) на основі виділених ринкових загроз та можливостей, та сильних і слабких сторін.

Таблиця 10.11 - Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	*Фактор конкурентноспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів у порівнянні з нашим підприємством*						
			-3	*-2*	*-1*	*0*	*+1*	*+2*	*+3*
1.	Використання ПЗ у вигляді сервісу	20			+				
2.	Простота інтерфейсу користувача	20	+						
3.	Висока точність класифікації	15		+					
4.	Хмарне розгортання	15					+		

5.	Вертикальне масштабування	10				+			

Отже, серед сильних сторін проекту можна виділити наступні: форма імплементації у вигляді сервісу, висока точність класифікації, можливість хмарного розгортання. Серед слабких сторін можна виділити можливість зміни тарифів провайдером хмарного розгортання, відсутність можливості горизонтального масштабування.

Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення. Наприклад: зниження доходів потенційних споживачів – фактор загрози, на основі якого можна зробити прогноз щодо посилення значущості цінового фактору при виборі товару та відповідно, – цінової конкуренції (а це вже – ринкова загроза).

У наступній таблиці буде проілюстровано SWOT-аналіз стартап-проекту, тобто його слабкі та сильні сторони, можливості та загрози виходу на ринок.

Таблиця 10.12 SWOT- аналіз стартап-проекту

Сильні сторони: форма імплементації у вигляді сервісу, висока точність класифікації, можливість хмарного розгортання	Слабкі сторони: можливість зміни тарифів провайдером хмарного розгортання на платні, відсутність можливості горизонтального масштабування
Можливості: зростання можливостей потенційних покупців, зниження довіри до конкурентів, попит, реформа навчальної програми для студентів-медиків, технологічний	Загрози: конкуренція, зміна потреб користувачів, зміна тарифів хмарного провайдера, вихід на ринок великої компанії, зміна податкової політики держави

На основі SWOT-аналізу розробляються альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проекту на ринок та орієнтовний

оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок. Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів.

Таблиця 10.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Навчання класифікаційної моделі з використанням програмного комплексу CUDA та відеокарт	90%	3 місяці
2.	Використання методів комп'ютерного зору для вирішення задачі класифікації	35%	8 місяців

З означених альтернатив обирається та, для якої: а) отримання ресурсів є більш простим та ймовірним; б) строки реалізації – більш стислими. Тому обираємо альтернативу (Навчання класифікаційної моделі з використанням програмного комплексу CUDA та відеокарт).

10.4 Розробка ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 14).

Таблиця 10.14 – Вибір цільових груп потенційних споживачів

№ п/п	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовни й попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1.	Лікарі-дерматологи (практикуючі та студенти)	Допомога в прийнятті правильного рішення при діагностуванн і хвороби.	Великий	Існує 3 конкуренти, які надають схожі, але більш вузькі і дорогі рішення.	Швидкодія, зручний користува-цький інтерфейс, точність класифікації, наявність REST API та

					можливість інтеграції з іншими сервісами.
2.	Пацієнти, що мають захворювання шкіри	Забезпечення можливості самостійної попередньої діагностики для пацієнтів.	Великий		Швидкодія, зручний користува-цький інтерфейс, точність класифікації, наявність REST API та можливість інтеграції з іншими сервісами.
Які цільові групи обрано: обираємо підприємства та дослідницькі центри					

За результатами аналізу потенційних груп споживачів (сегментів) автори ідеї обирають цільові групи, для яких вони пропонуватимуть свій товар, та визначають стратегію охоплення ринку. Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку. За М. Портером, існують три базові стратегії розвитку, що відрізняються за ступенем охоплення цільового ринку та типом конкурентної переваги, що має бути реалізована на ринку (за витратами або визначними якостями товару). Отже, проілюструвати базову стратегію розвитку можна у виляді Таблиці 4.15.

Таблиця 10.15 – Визначення базової стратегії розвитку

№ п/п	*Обрана альтернатива розвитку проекту*	*Стратегія охоплення ринку*	*Ключові конкурентоспромо жні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку*
1.	Навчання класифікаційної моделі за	Ринкове позиціонування	Швидкодія, точність класифікації.	Диференціація

	допомогою програмної платформи CUDA на базі сучасних відеокарт, призначених для навчання нейронних мереж.			

Було обрано таку альтернативу розвитку як навчання класифікаційної моделі за допомогою програмної платформи CUDA на базі сучасних відеокарт, призначених для навчання нейронних мереж. Адже завдяки цим технологіям можна досягнути високої точності класифікації та ключових конкурентноспроможних позицій кінцевого продукту.

Таблиця 10.16 - Визначення базової стратегії конкурентної поведінки

№ п/п	*Чи є проект «першопрохідцем» на ринку?*	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки*
1.	Ні	Так	Буде, а саме: основною задачею є розробка ПЗ з використанням згорткової нейронної мережі (конкуренти 1, 2, 3), форма виконання - веб-сервіс (конкурент 2)	Зайняття конкурентної ніші

Отже, було визначено базову стратегію конкурентної поведінки як зайняття конкурентної ніші.

Визначимо стратегію позиціонування у Таблиці 10.17, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 10.17 - Визначення стратегії позиціонування

№ п/п	*Вимоги до товару цільової аудиторії*	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап- проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)*
1.	Швидкодія, зручний користувацький інтерфейс, точність класифікації, наявність REST API та можливість інтеграції з іншими сервісами.	Диференціація	Висока точність класифікації, можливість інтеграції в уже існуючі системи завдяки REST API, зручне хмарне розгортання.	Висока точність класифікації, інтеграція, хмарне розгортання.

Отже, було вибрано такі асоціації, які мають сформувати комплексну позицію власного проекту: висока точність класифікації, інтеграція (адже завдяки REST API сервіс просто інтегрувати у існуючі системи), хмарне розгортання.

10.5 Розробка маркетингової програм

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у табл. 10.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 10.18 - Визначення ключових переваг концепції потенційного товару

№ п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)*
1.	Швидкодія	ПЗ працює досить швидко, результат можна отримати до 10 мс	Перевага у швидкості
2.	Простота користувацького інтерфейсу	Простота роботи додатку	Користувачі мають зручний інтерфейс для взаємодії з ПЗ
3.	Наявність REST API	Можливість інтеграції в уже існуючі системи	Можливість інтеграції в уже існуючі системи

Отже бачимо, що проект має ключові переваги перед конкурентами, які повністю відповідають потребам цільової аудиторії. Висока точність класифікації захворювань та швидкість роботи з додатком є основними перевагами концепції товару, а наявність REST API надає можливість інтеграції із сторонніми сервісами.

Далі у Таблиці 10.19 проілюстрована трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання.

Таблиця 10.19 - Опис трьох рівнів моделі товару

<table>
<tr><th>Рівні товару</th><th colspan="3">Сутність та складові</th></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Сервіс автоматизованої класифікації захворювань шкіри (як доброякісних, так і злоякісних) на базі зображення ураженої ділянки тіла.</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>Форма реалізації ПЗ у вигляді сервісу.
Висока точність класифікації.
Наявність REST API для інтеграції зі сторонніми сервісами.
Висока швидкість роботи сервісу.</td><td>1.Нм
2.Нм
3.Нм
4.Нм</td><td>1.Технологічна
2.Технологічна
3.Технологічна
4.Технологічна</td></tr>
<tr><td colspan="3">Якість: згідно до стандарту ISO 4444 буде проведено тестування</td></tr>
<tr><td colspan="3">Маркування відсутнє</td></tr>
<tr><td colspan="3">Моя компанія: “Skin diseases”</td></tr>
<tr><td rowspan="2">III. Товар із підкріплюнням</td><td colspan="3">1-місячна пробна безкоштовна версія</td></tr>
<tr><td colspan="3">Постійна підтримка для користувачів</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: патент</td></tr>
</table>

Було описано три рівні моделі товару, з чого можна зробити висновок, що основні властивості товару у реальному виконанні є нематеріальними та

технологічними. Також було надано сутність та складові товару у задумці та товару з підкріпленням.

Після формування маркетингової моделі товару слід особливо відмітити – чим саме проект буде захищено від копіювання. У даному випадку найбільш вірогідним гарантом буде патент.

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 10.20). Аналіз проводиться експертним методом.

Таблиця 10.20 - Визначення меж встановлення ціни

№ п/п	*Рівень цін на товари-замінники, грн.*	*Рівень цін на товари-аналоги, грн.*	*Рівень доходів цільової групи споживачів, грн.*	*Верхня та нижня межі встановлення ціни на товар/послугу, грн.*
1.	45000	38000	150000	35000-40000

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 10.21).

Таблиця 10.21 - Формування системи збуту

№ п/п	*Специфіка закупівельної поведінки цільових клієнтів*	*Функції збуту, які має виконувати постачальник товару*	*Глибина каналу збуту*	*Оптимальна система збуту*
1.	Придбання підписки та оплата щомісячних внесків для продовження ліцензії	Продаж	0 (напряму), 1 (через одного посередника)	Власна та через посередників

Отже, система приноситиме прибуток завдяки щомісячним внескам для подовження ліцензії та придбанням підписок, продаж будк виконуватись напряму або через одного посередника.

Останньою складовою маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 10.22).

Таблиця 10.22 - Концепція маркетингових комунікацій

№ п/п	*Специфіка поведінки цільових клієнтів*	*Канали комунікацій, якими користуютьс я цільові клієнти*	*Ключові позиції, обрані для позиціону вання*	*Завдання рекламного повідомлення*	*Концепція рекламного звернення*
1.	Придбання ліцензії на користування в мережі Інтернет, щомісячне її продовження, користування сервісом у хмарі або ж на власних серверах.	Інтернет	Висока точність класифіка ції, інтеграція , хмарне розгортан ня.	Показати переваги сервісу, у тому числі і перед конкурентами	Демо-ролик із використання, рекламні оголошення на популярних сайтах.

Отже, в Таблиці 10.22 наведено концепцію маркетингових комунікацій, було визначено, що придбання ліцензії на користування буде здійснюватись в мережі Інтернет, необхідним буде щомісячне її продовження, користування сервісом можливе у хмарі або ж на власних серверах.

10.6 Висновки

Згідно до проведених досліджень існує можливість ринкової комерціалізації проекту. Також, варто відмітити, що існують перспективи впровадження з огляду на потенційні групи клієнтів, бар’єри входження не є високими, а проект має дві значні переваги перед конкурентами. Для успішного виконання проекту необхідно реалізувати сервіс із використанням технологій Python, Tensorflow, CUDA. Для успішного виходу на ринок у продукту повинні бути наступні характеристики:

- Висока точність класифікації
- Швидкість роботи сервісу
- Наявність універсального REST API та можливість інтеграції зі сторонніми сервісами
- Можливість зручного хмарного розгортання

В рамках даного дослідження були розраховані основні фінансово-економічні показники проекту, а також проведений менеджмент потенційних ризиків. Проаналізувавши отримані результати, можна зробити висновок, що подальша імплементація є доцільною.

Було визначено такі сильні сторони: форма імплементації у вигляді сервісу, висока точність класифікації, можливість хмарного розгортання. Серед слабких сторін можна виділити можливість зміни тарифів провайдером хмарного розгортання на платні, відсутність можливості горизонтального масштабування.

Можливості для виходу на ринок включають попит серед лікарів-дерматологів на інструменти автоматизованої класифікації шкірних захворювань, стрімке зростання росту ринку, обслуговування додаткових груп споживачів, розширення асортименту можливих послуг. Наявні такі фактори загроз: конкуренція, зміна потреб користувачів, зміна тарифів провайдера хмарного розгортання (а саме підвищення орендної плати за використання інфраструктури), надходження на ринок альтернативних продуктів, уповільнення росту ринку.

ВИСНОВКИ

В рамках дипломного проекту було розглянуто теоретичний базис моделі багаторівневого навчання для побудови автоматизованої діагностичної системи. Дана система служить для вирішення задачі класифікації шкірних захворювань на основі зображень враженої ділянки тіла людини.

Наукова новизна роботи полягає в створенні прототипу автоматизованої діагностичної системи на основі багаторівневого навчання для вирішення задачі класифікації захворювань шкіри, а також апробації отриманого прототипу на практиці з використанням вибірки даних, представленої міжнародною спільнотою з цифрової обробки зображень шкіри ISDIS (13791 зображень уражених ділянок шкіри, зроблених за допомогою дермоскопу).

Також було досліджено архітектуру згорткових нейронних мереж для аналізу зображень та визначено основні способи боротьби з можливими проблемами (такими як затухаючих градієнт, проблема перенавчання). Виконано огляд програмних інструментів для практичної побудови та навчання нейронних мереж та обґрунтовано вибір інструменту Tensorflow. Класифікаційна модель була реалізована у вигляді графу обчислень за допомогою мови програмування Python та модуля Tensorflow.

В якості функції помилки для діагностики точності отриманої моделі використовувалась перехресна ентропія. Вибірку було розділено на тестову та валідаційну і для кожної епохи навчання було обчислено: значення точності класифікаційної моделі для тестової вибірки та для валідаційної, значення функції помилки. В результаті навчання отримано класифікаційну модель, яка на валідаційній вибірці демонструє високу точність, а саме 81.2%, що підтверджує ефективність використання алгоритмів багаторівневого навчання для допомоги в діагностуванні хвороб лікарями-дерматологами, а також для полегшення навчання студентам-дерматологам.

Отримані результати можуть використовуватись у майбутніх дослідженнях за напрямком покращення запропонованого прототипу та класифікаційної моделі, враховуючи переваги та недоліки даних результатів. Також даний прототип може бути використаний для покращення результатів роботи існуючих діагностичних систем.

Серед основних проблем, що виникли під час тренування даної згорткової мережі – це відсутність доступу до потужних дискретних відеокарт для запуску навчання моделі на базі платформи CUDA. А також низька пропускна швидкість каналу призвела до проблем з часом обробки вхідних зображень навчального набору.

ПЕРЕЛІК ПОСИЛАНЬ

1. Форсайт Д.А., Понс Ж. Компьютерное зрение. Современный подход. - М.: Вильямс, 2004. – 928 с.

2. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. - М.: Мир, 1992. – 184 с.

3. Хайкин С. Нейронные сети: полный курс, 2-е издание. – М.: Вильямс, 2008. – 1103 с.

4. Флах П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. – М.: ДМК Пресс, 2015. – 400 с.

5. Schmidhuber, J. Deep Learning in Neural Networks: An Overview / J. Schmidhuber // Neural Networks. — 2015. — Vol. 61. — P. 85–117.

6. Rosenblatt, F. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms / F. Rosenblatt. — Buffalo, N.Y. : Cornell Aeronautical Laboratory, 1961. — xviii, 622 p. — (Cornell Aeronautical Laboratory; Report no. VG-1196- G-8).

7. Akymov V. Deep learning algorithms for classification of skin diseases / Vadym Akymov.// International Scientific Journal “Internauka”.–2018.–№9.

8. Акимов В. Алгоритми багаторівневого навчання для класифікації захворювань шкіри. / Вадим Акимов.// Матеріали XX всеукраїнської науково – практичної студентської конференції, 21 травня 2018, Київ, Україна: матеріали. – К. : НТУУ «КПІ», 2018. – С. 121-122.

9. J. Arroyo and B. Zapirain. Automated detection of melanoma in dermoscopic images. In J. Scharcanski and M. E. Celebi, editors, Computer Vision Techniques for the Diagnosis of Skin Cancer, Series in BioEngineering, pages 139–192. Springer Berlin Heidelberg, 2014.

10. A.A.Cruz-Roa,J.E.A.Ovalle,A.Madabhushi,andF.A.G. Osorio. A deep learning architecture for image representation, visual interpretability and automated basal-cell carci- noma cancer detection. In Medical Image Computing

and Computer-Assisted Intervention–MICCAI 2013, pages 403– 410. Springer, 2013.

11. K. Simonyan and A. Zisserman. Very deep convolu- tional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.

12. A.MasoodandA.AliAl-Jumaily.Computeraideddiagnos-
tic support system for skin cancer: A review of techniques and algorithms. International Journal of Biomedical Imag- ing, 2013, 2013.

13. Krizhevsky, A. ImageNet Classification with Deep Convolutional Neural Networks / A. Krizhevsky, I. Sutskever, G. E. Hinton // Advances in Neural Information Processing Systems 25. — 2012. — P. 1106–1114.

14. Deng, L. Deep Learning: Methods and Applications / L. Deng, D. Yu // Foundations and Trends in Signal Processing. — 2014. — Vol. 7 (3–4), P. 197–387.

15. Bishop C. M. Pattern Recognition and Machine Learning. — New York : Springer, 2006. — xx, 738 p., ill.

16. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. Погружение в мир нейронных сетей. 978-5-496-02536-2 изд. СПб. Питер: ООО Издательство "Питер", 2018. 480 с.

17. Nikhil Buduma Fundamentals of Deep Learning. Designing next-generation machine intelligence algorithms. 978-1-491-92561-4 O'Reilly Media, 2017. 277 с.

18. Francois Chollet Deep Learning with Python. 978-1617294433 изд. Manning Publications, December 22, 2017. 384 с.

19. Lu, L., Zheng, Y., Carneiro, G., Yang, L. Deep Learning and Convolutional Neural Networks for Medical Image Computing. 978-3-319-42998-4 изд. Springer International Publishing, 2017. 326 с.

20. Google Research Team. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv:1603.04467 [cs.DC], 2016.

21. Docker, Docker Documentation. – https://docs.docker.com/

ДОДАТКИ

Додаток А

Вихідний код парсеру зображень.

```
import urllib.request as request
import numpy as np
import math
import matplotlib
# for MAC OS
matplotlib.use('TkAgg')
import tensorflow as tf
import matplotlib.pyplot as plt
import json
import os.path


# ISIC images count
IMAGES_COUNT = 13791
# imageCount = 1000
API_BASE_URL = "https://isic-archive.com/api/v1"
X_SIZE = 250
Y_SIZE = 250


def read_images_metadata():
    # list of image ids
    images_metadata = []
    offset = 0
    limit = 50
    for i in range(1, math.ceil(IMAGES_COUNT / limit) + 1):
        print("From: ", offset, " to: ", offset + limit)
        part_of_data = request.urlopen(API_BASE_URL +
"/image?limit=50&sort=name&sortdir=1&detail=true&offset=" + str(offset)).read()
        images_metadata.extend(json.loads(part_of_data))
        offset += limit
    return images_metadata


def retrieve_image_ids(metadata_list):
    images_ids = []
    for item in metadata_list:
        images_ids.append(item["_id"])
    return images_ids


def retrieve_diagnosis(metadata_list):
    diagnosis_list = []
    for item in metadata_list:
        try:
            diagnosis_list.append(item['meta']['clinical']['diagnosis'])
        except KeyError:
```

```
            diagnosis_list.append("")
            print("No diagnosis field in: ", item)
    return diagnosis_list


imageIds = []
# last rows can be without diagnose
diagnosis = []
# check whether numpy array with image ids exists
if os.path.exists("ids.npy") and os.path.exists("diagnosis.npy"):
    print("Image ids was loaded from numpy array")
    imageIds = np.load("ids.npy")
    diagnosis = np.load("diagnosis.npy")
else:
    print("Image ids was loaded from REST")
    # read images metaData
    imagesMetadata = read_images_metadata()
    imageIds = np.array(retrieve_image_ids(imagesMetadata))
    np.save("ids", imageIds)
    diagnosis = np.array(retrieve_diagnosis(imagesMetadata))
    np.save("diagnosis", diagnosis)

# counter of loaded images
loaded_counter = 0
for image_index in range(loaded_counter, IMAGES_COUNT + 1):
    # read data from ISIC REST API
    imageData = request.urlopen(API_BASE_URL + "/image/" + imageIds[image_index] +
"/download").read()

    # open tensorflow session
    s = tf.Session()
    imageTensor = tf.image.decode_jpeg(imageData, channels=3)
    resized_image = tf.cast(tf.image.resize_images(imageTensor, [X_SIZE, Y_SIZE]), tf.uint8)
    array = s.run(resized_image)
    np.save("images/img" + str(loaded_counter), array)
    print("img", loaded_counter)
    print(array)
    loaded_counter += 1
```

Додаток Б

Вихідний код обчислювального графу нейронної мережі.

```
import tensorflow as tf
import numpy as np

# total number of output classes
N_CLASSES = 3
# Dropout, probability to keep units
DROP_OUT_RATE = 0.75
BATCH_SIZE = 128
LEARNING_RATE = 0.001
```

```
def conv_net(images, weights, biases, labels, mode):
    # tensorboard writer
    writer = tf.summary.FileWriter("tensorboard/1")
    # create session and initialize variables
    s = tf.Session()
    # [Convolution #1]
    with tf.variable_scope('conv1') as scope:
        filters_count = 32
        kernel_size = 5

        conv1 = tf.layers.conv2d(
            inputs=images,
            filters=filters_count,
            kernel_size=kernel_size,
            activation=tf.nn.relu,
            kernel_initializer=tf.random_normal_initializer(),
            padding='SAME')

        s.run(tf.global_variables_initializer())

        # write input images summary to tensorboard
        image_summary_tensor = tf.summary.image("input_image", images, 3)
        writer.add_summary(s.run(image_summary_tensor))

        # #write conv1 to tensorboard
        # pool_image_summary_tensor = tf.summary.image("conv1_image", tf.reshape(conv1, [32,
250, 250, 1], name="conv1"), 32)
        # writer.add_summary(s.run(pool_image_summary_tensor))

    with tf.variable_scope('pool1') as scope:
        pool1 = tf.layers.max_pooling2d(
            inputs=conv1,
            pool_size=[2, 2],
            strides=2
        )

        # write input images summary to tensorboard
        pool1_image_summary_tensor = tf.summary.image("pool1_feature_image",
tf.reshape(pool1, [32, 125, 125, 1], name="pool1"), 32)
        writer.add_summary(s.run(pool1_image_summary_tensor))

    # [Convolution #2]
    with tf.variable_scope('conv2') as scope:
        filters_count = 64
        kernel_size = 5

        conv2 = tf.layers.conv2d(
            inputs=pool1,
            filters=filters_count,
            kernel_size=kernel_size,
            activation=tf.nn.relu,
```

```
            kernel_initializer=tf.random_normal_initializer(),
            padding='SAME')

        s.run(tf.global_variables_initializer())

    with tf.variable_scope('pool2') as scope:
        pool2 = tf.layers.max_pooling2d(
            inputs=conv2,
            pool_size=[2, 2],
            strides=2
        )

        # write input images summary to tensorboard
        pool2_image_summary_tensor = tf.summary.image("pool2_feature_image",
                                        tf.reshape(pool2, [64, 62, 62, 1], name="pool2"), 64)
        writer.add_summary(s.run(pool2_image_summary_tensor))

    # Dense Layer
    with tf.variable_scope('dense') as scope:
        pool2_flat = tf.reshape(pool2, [-1, 64*62*62])
        dense = tf.layers.dense(inputs=pool2_flat, units=64*62*62, activation=tf.nn.relu)
        dropout = tf.layers.dropout(
            inputs=dense, rate=DROP_OUT_RATE, training=mode ==
tf.estimator.ModeKeys.TRAIN)
        # Logits Layer
        logits = tf.layers.dense(inputs=dropout, units=10)

    # Calculate Loss (for both TRAIN and EVAL modes)
    loss = tf.losses.sparse_softmax_cross_entropy(labels=labels, logits=logits)
    # Configure the Training Op (for TRAIN mode)
    if mode == tf.estimator.ModeKeys.TRAIN:
        optimizer = tf.train.GradientDescentOptimizer(learning_rate=LEARNING_RATE)
        train_op = optimizer.minimize(
            loss=loss,
            global_step=tf.train.get_global_step())
        return tf.estimator.EstimatorSpec(mode=mode, loss=loss, train_op=train_op)

    writer.add_graph(s.graph)
    s.run(logits)



image = np.load("images/img0.npy")
t = tf.convert_to_tensor(image, dtype=tf.float32, name="initial_image")
reshaped_image = tf.reshape(t, [-1, 250, 250, 3], name="image")
# r1 = s1.run(input)
# # print(r1)
# # print(r1.shape)

conv_net(reshaped_image, None, None, None, tf.estimator.ModeKeys.TRAIN)
```